ChaosPro

Martin Pfingstl

COLLABORATORS			
	TITLE : ChaosPro		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Martin Pfingstl	February 12, 2023	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

ChaosPro

1.1 Contents

Contents

1.2 Preface

I. Introduction

1.1 Preface

What? Yet another fractal-creating program ??

On the one side one could think that there are enough programs for creating fractals
on the Amiga. But if you are looking at other platforms e.g. IBM PC with FractInt
from the Stone Soap Group, then everybody is realizing quickly that the programs on
the Amiga aren't as powerful as they seem in the first way: None of them is able to
create as many different fractal types as FractInt and none of them allows changing
as many parameters as FractInt. Well, FractInt surely isn't perfect, but it's good
enough for a few ideas...

If you have a look at this program, then you will notice, that ChaosPro is quite powerful and huge.
Constantly the users of ChaosPro have wondered themselves, why ChaosPro is Public Domain. Many other,
much smaller, much buggier, less powerful programs are Shareware. I thought about all this for some time, following some
remarks:

1. If I want to earn money by programming, then for sure I won't have written a fractal generating program
for the Amiga.

2. What would happen, if ChaosPro would be Shareware?
Other authors have tried to write a fractal generating program (or another program) and then were quite angry, because
almost nobody has sent the Shareware fee for their absolutely great program. They were so angry, that they
decided to not further develop the program.

```
Well, I always ask myself, what IQ these people must have. Who should pay the ↩
    Shareware fee for
a fractal generating program? One doesn't use such a program on a regular basis, ↩
    it's of no use, one just starts it, because
one has some spare time left and just wants to see and calculate some nice ↩
    pictures, and that's
all...

3. So I decided, to renounce the Shareware fee from about 5 people. That is so few ↩
     money, that it isn't worth the effort.
So ChaosPro is further Public Domain, that's the logical conclusion. This way ↩
    people are more willing to write
mails to me and to suggest further enhancements, and that's what I want. This way ↩
    I feel, that people are using
ChaosPro, so it makes more fun to enhance it.
```

1.3 Why should I use this program?

```
1.2 Why should I use this program?

In other words: What are the advantages of this program over all the other fractal ↩
     creating
programs? Well, if you were content with the other fractalprograms and never ↩
    reached the point,
where these programs weren't able to satisfiy your needs, then I think it's ↩
    probable, that another
program would be the better solution for you. This program seems to be a bit ↩
    confusing, because it
has many parameters, i.e. you can make several mistakes, and this can be somehow
discouraging.
If you just want to calculate a few fractal pictures, then this program surely is ↩
    a bit
too large for you. You don't buy Brilliance or DPaint IV AGA just to paint some ↩
    icons,
do you?

Following a few features of the program:

(Inspired by Mand2000Demo, FractInt, MisterM, MandelMania, Fractal Dynamics, ↩
    Slicer,
MultiFractals, MandelMountains, Fractal V1.3, MandelPlot 24, Mandelsquare, ↩
    SmartFractal,
LyapunoviaV1.5, CloudsAGA, KFP and FractalUniverse)

- Multiwindowing
All fractals are drawn in windows, which you can easily enlarge or smallen by ↩
    using the
sizegadget.

- Multitasking
For the calculation of each fractal a separate task is created, i.e. you can ↩
    calculte
several fractals at the same time.

- Realtime-effects
```

Changes of parameters have immediate effects.

- Click and Zoom
Just doubleclick at a point and you zoom in and this point...

- Move the area around
The area of the complex numberplane, from which the fractal is calculated, can be moved around while calculating the fractal. Just click and drag it with the mouse or use the cursor-keys or the joystick in port 2.

- Systemconform
According to my betatesters the program runs perfectly on:
 - Picasso
 - Piccolo
 - GVP EGS110/24
 - GVP Spectrum
 - ECS/OCS
 - AGA
 - Merlin

It runs from OS2.0 upto OS3.1, a screenmode-requester is used to enable to use all resolutions.

- Formula editor
For all of you, who want to try their own formula.

- Several fractaltypes
 - Juliaset
 - Mandelbrotset
 - Bifurcationdiagrams (Verhulst)
 - Dynamic Systems
 - Plasma
 - Lyapunov-Spaces

- Parameter
Dependent on the fractaltype upto 3 parameterwindows exist.

- Logical Userinterface
The worst example for a program, which normally mustn't exist: FractInt on a PC. There exists an Amiga-version by Terje Pedersen (email: terjepe@stud.cs.uit.no), which is a bit better (it uses MUI).

- 3D-Transformations
There exist 3 more windows with parameters just for 3D. Of course I must admit, that the right parametervalues are a bit difficult to find. But the multitasking of the program (you see immediately the result of a change of a parameter) helps really. You see quickly, whether the value suits or not.

- Animations
Not only simple Zoom-in-Movies, but also Zoom-out-Movies, or any other animation based on a parameter,

which continually changes its value. Of course, more than one parameter may change its value.
How about a 3D-Anim Zoom-in-Movie into a juliaset, whose parametervalue 'c' changes and the light
moves around?

- 24 Bit
Fractals may be saved in 24 bit.

- Online-Help
Of course contextsensitive ;-)

- Locale-support
Why not?

- Arexx-Interface
Sorry, it made so much fun, I weren't able to stop ;-)

- and some other small features - really small;-)
+ Filename-Multiselect
+ Menu-Multiselect
+ Colorwheel under OS3.0 while editing a palette
+ Pictures can be saved into the clipboard
+ Fontsensitivity

...and a little bit more...

1.4 Requirements

1.3 Requirements

When I was writing this program, I often had to decide whether I should leave a feature aside in order
to allow this program to run even on a badly equipped amiga. Because I think that in
the year 1995 it's time to realize that a 7.09 Mhz-68000 Amiga isn't state of the art,
I've decided that this program will run only on better Amigas.

The program needs at least an 68020 with a mathematical coprocessor. Due to the internal
multitasking the screen is quickly filled with many windows. So a higher resolution which
offers more place is recommended. These many windows also don't increase the speed
of the system, so a fast Amiga is a good thing...

Because this program has many features, it needs a lot of memory. You should have at
least 2 MB RAM, then you can test this program.

Of course, the version of the operating system must be at least 2.0.

1.5 Installation

```
                1.4 Installation and Deinstallation

Installation is made by the 'Installer' from Commodore.
If you want to make it by hand, then here you go:
1. Copy reqtools.library to the logical directory libs: , if it isn't already ↩
   there.
You need at least V38 or above.
2. Copy the directory ChaosPro/ and all the subdirectories to your desired place.
3. Copy the contents of the fonts-directory into your logical FONTS: directory..
.

That's all. Installation is then finished. To adjust the program to your system, ↩
   start
the
                preferencesprogram
                .

If you are looking at the file 'english.guide', don't wonder, why AmigaGuide doesn ↩
   't find all
nodes in this file. It will be translated to the file ChaosPro.guide by the ↩
   preferencesprogram.
While translating unknown links to nodes will be solved 'magically'.

In order to deinstall ChaosPro, just delete the whole ChaosPro-directory with all ↩
   files in it. Please have a look into
the directory libs:, too. Perhaps there is - for some unknown reason - the library ↩
    ChaosPro.library. Delete it...
Normally ChaosPro doesn't copy any files over your hard disk...
```

1.6 Author

```
1.5 Author

Address:
        Martin Pfingstl
        Dorfen 16 1/5
        84508 Burgkirchen
        Call.: holiday: 0 86 79 / 62 41 semester: 089 / 28 44 91
        ( Germany )

        EMail: pfingstl@informatik.tu-muenchen.de

Many thanks in advance for:
- bugreports
- new ideas
- comments
- no registrations ;-) (ChaosPro is Public Domain...)
```

1.7 Concept

```
                II. Programdescription

2.1 Concept

The concept isn't new, but very useful: Multitasking and multiwindowing over and ↩
   over.
That means, you can calculate as many fractals at the same time as you wish (and ↩
   as your memory allows).
You can work on a
                palette
                 and calculate an animation, while
an ARexx-program controls some other fractals beeing calculated.
You can read in the online-help, and without closing this window you can ↩
   experience
in another window what you have read.
```

1.8 PicTask

```
                2.2 The Various Windows

2.2.1 PicTask-Window

Fractal Pictures
-  In the viewwindow with the headline 'Fractal Pictures' all to the program at ↩
   the moment known
fractal pictures are displayed. Every entry has a corresponding data structure ↩
   which contains all parameters
needed to calculate a picture of the fractal. Whenever you start the program, it ↩
   examines the directory
'FractPic' and loads automatically all fractals it finds there.
See Chapter 2.6
                Programdirectories
                .

Name of a Picture
-  Directly below the right listview there is a string gadget, in which you can ↩
   edit the name.
In order to take effect you have to press the return key.

Clear Picture
-  Press this gadget to delete the chosen picture.

Calculate Picture
- If you click onto this gadget, the active entry appears additionally in the left ↩
    listview.
A window is opened and a task created which then calculates the corresponding ↩
   fractal in the
newly opened window. This task runs with a priority of 1 less than the control- ↩
   task. So
controlling functions slow down the speed of calculating the fractal.
```

Duplicate Picture
- Under some circumstances somebody wants to change a few parameters of a fractal without changing the old fractal. This gadget duplicates the active entry so a new entry is added and so you can change values and have the old fractal left.

Close Windows
- This gadget closes all windows, which belong to the active fractal. To be more precise, it deletes the task, the fractalwindows and all its parameterwindows.

Setting Previewwidth/-height
- With these gadgets the size of the area, which is calculated first, is defined. The area is placed in the middle of the window, where in all probability the most interesting part of the fractal beeing created is hidden. If you set unlogical values, then so preview is calculated. Preview isn't possible with all kinds of fractals, so it's possible, that these values take no effect.

Picture settings

3D-Buffertype
Julia- and Mandelbrotsets can be transformed into the 3rd dimension. In order to get good looking pictures, it's possible, to allocate a buffer for the results of the 3D-transformation. This enables it to save 3D-pictures in 24 Bit. Additionally there exist 2 more gadgets in the 3D-parameterwindow number 3, which have some influence on the appearance of the 3D-image. They control, how the incoming light changes the original color of the 3D-fractal.

Buffertype
- There are 3 different types for the buffer:
1. No buffer: This choice uses of course less memory than the other ones. But on the other hand you can't calculate 3D-views of the fractal, because the routines for this force the availability of a buffer. Also saving s IFF-ILBM-picture is only possible in the depth of the screen.
2. 16Bit-Int: Here for every point a word (16bit) is resersed, in which the calculated value is put in. Here you can choose a 3D-view. Additionally it's possible to save the fractal in any depth from 3 to 8 planes and in 24 bit.
3. 32 Bit IEEE Single Precision-Buffer (for people with too much memory): Here for every pixel a whole longword is reserved, in which the exact value of the point is placed in the IEEESP-format. This choice makes it possible, to save the inside area of the julia-/ mandelbrotset in real 24Bit.

Windows
2 choices:

```
- 1 Window: If a 3D-view of the fractal should be drawn, then it's drawed in the same window as
the 2D-fractal. The 2D-fractal will be overdrawn.
- 2 Windows: If a 3D-view of the fractal should be drawn, then a second window will
be opened for this purpose.

3D
And again 2 choices:
- No 3D-Picture: Only the (2D)fractal will be drawn.
- 3D-Picture: After drawing the 2D-fractal all data will be interpreted as heights and
a 3D-view will be drawn.

Choice of the palette
Whenever a new window gets active, the program tries to find out, what palette should
be used. For this it looks for the fractal, which the window belongs to and sets the
corresponding palette.
The program always has a global palette. Additionally there exist two extries in the fractalstructure
for palettenames. The one name is ment for the palette to be used for the 2D- fractal,
the other name for the palette to be used for the 3D-fractal.
In order to control the behaviour of the program, when a new window is activated, there
exist 2 gadgets. If the checkboxgadget is checked, always the global palette is used,
independent from the fractal and its 2 own palettenames. This mode is mainly ment for such
people like me, who get confused, if suddenly another palette is used, when a new window
is activated (I use 'SunWindow' - created by Bernhard Scholz - this is advertising ... - for
autoactivating windows...)
If the checkboxgadget isn't checked, then the cyclegadget right beneath determines the
palette to be used for the fractal. 'Own palette' effects, that the palette is used,
which is defined in the fractalstructure. 'Global palette' effects, that the global palette is used, whenever a
window is activated, which belongs to this fractal.

If somebody wants to change the global palette, then he only has to wait, until
the global palette is used. Then he can use the palettewindow in order to change it.
This is exactly what one would expect. So don't get confused.
```

1.9 Palettes

2.2.2 Palettewindow

The palettewindow contains all palettes, which the program has found in the directory
ChaosPro/Palette/ at startup. There may be whole pictures, in that case the colorchunk is
filtered out and added as palette.

At the beginning the palette called 'DefaultMap' is the active one. If you prefer
another palette, then place it in the directory Palette and change it's name to
'DefaultMap'. If no palette with this name is existant, then the very first palette
is the active one.

If you want to set another colortable, then simply click on the desired entry. The change
takes effect immediatly. In order to maintain the 3D-effect of the graphical environment
the colors 0 to 3 aren't affected.

Palettename
- To change the name of the active palette, change it in the string gadget and
leave it with pressing the return key.

Edit Palette
- If you want to edit the colors of the active palette, click onto this gadget.
2 windows are opened, the one which displayes the current values and allows you to
take some actions and the other which shows you the varous colors of the active
palette with a palette gadget. If 256 colors are available on the fractalscreen,
the windows are opened on it, otherwise a new screen called colorscreen is opened as
defined by the
preferences-program
See Chapter 2.1.3
EditPalettewindow
load and save Palettes
- With these gadgets you can load and save palettes. If you would overwrite an existing
palette, you are pleased to confirm your action. When loading a palette, of course file-multiselect
is supported, so you can load many palettes at once. Starting with V2.0 ChaosPro is also able to load palettes, which were created for FractInt. These
files normally have the suffix '.map'.

Clear and Duplicate a Palette
- Should I really explain these ?

Coloroffset
- This gadget defines the colornumber of the palette, at which the palette is used.
Example: Somebody has a screen with 32 colors, he sets this value to 30. The screen
now gets the colors of the palette beginning at number 30 (screennumber 4) upto
number 57 (screennumber 31). If you change this value continual, then you'll get
a colorcyling-effect.

Skip
- If this value is set to, as example, 2, then only every 2nd color of the palette is used for

the screen. This makes sense for palettes, which actually are made for 256-color- ↩
screens
and should now be used on, as example, a 32-color-screen. There you could set the ↩
skip-value to 8
and so using only every 8th color of the palette. So you'll get an imagination of ↩
what
the palette would look like on a 256-color-screen.

1.10 Editing a Palette

2.2.3 EditPalettewindow

Actually there are 3 windows. One window for the actions and for displaying the ↩
values of a
specified color the other for showing the whole palette. The third eventually for ↩
the colorwheel and the
gradientslider

Colorarea
- Not everybody has the possibility to display the whole palette. Due to this, the ↩
colorarea-gadget
is existant. It shows with the size and position of the bar, how many colors out ↩
of which area are
currently displayed in the other window. If you move it around, then automatically ↩
the colors in the other
window are actualized according to the new position of the bar.

Colornumber
- This gadget shows the current registernumber. If you change it, then the RGB- ↩
and
HSV-values are updated.

The RGB- and HSV-Slider
- These sliders change the colorvalues. If you change one value in the colormodel ↩
, the others
in the other model are updated according to changes.

Copy, Exchange, Spread
- 'Copy' copies the active color to the position, which the user next clicks onto ↩
, 'Exchange' exchanges
the colors and 'Spread' makes a smooth change from the one color to the next ↩
active one.

Cycling-Mode
- This mode is somehow inconvenient. If you click onto this gadget, then you are ↩
in a mode,
in which you can exactly define, what colors should be affected by colorcycling.
This is useful for the mandelbrot, if you want to cycle only the area outside,
because the inside area is colored black. All visible colors are affected by the ↩
colorcycling,
all grey blinking colors, aren't affected. Click on a color, and the state of a ↩
color is changed.
The 3 gadgets 'All','None' and 'Invert' do exactly what someone expects: 'All' ↩
lets

```
all colors take part on colorcycling, 'None' no color and 'Invert' inverts the ↩
    state
of all colors.

Functions affecting an Area
- 'Shrink to' and 'Shrink'
These both gadgets make it possible, to shrink the palette to less colors. To do ↩
    so,
you choose with the slider the number of planes, and with 'Shrink' you execute the
action.
- 'Invert Area'
This gadget allows you, to invert an area. To do so, you click onto the gadget, ↩
    then on
the first color of the area, then on the last color of the area.
- 'Copy Area'
Click onto the gadget, then onto the first color, then onto the last color and ↩
    then
onto the first color of the destination area. Overlaying, overflowing etc. areas ↩
    are
affected correctly.

Colors of a Palette
- This window shows the colors of a palette. It has a size-gadget, so you can ↩
    adjust it's size
to your preferred size.

Colorwheel
People, who have OS3.0, can use the colorwheel, in which they can pretty much ↩
    intuitively
choose the colors. In order to get the colorwheel, one has to add the tooltype ↩
    COLORWHEEL.
Because the colorwheel needs half of the available colors on the colorscreen, this ↩
     method
exists to enable or disable the colorwheel.
```

1.11 Animationwindows

2.2.4 Animationwindows

```
With these windows you can calculate animations. For that purpose fractals, which ↩
    may differ
only in continual changeable parametervalues, are defined as keys. At calcalation ↩
    time
the intermediate values between the different parameters of two keys are ↩
    calculated ( that's
why it's called "continual changeable parametervalues" ) and as data structure ↩
    given to
a fractal task which interprets and acts accordingly.
This is a really good method, I think. You can e.g. the parameter c of the ↩
    standard
juliaset in a continual way change, and at the same time you can change the area ↩
    values
and the number of iterations. The result is a zoom-movie into an altering juliaset ↩
    , a zoom and morph at
the same time...
```

Now I come to a description of the gadgets:

Fractal Pictures
- Here the fractal pictures of the
PicTask-Window
are displayed
again for your convenience.

Anim Keys
- That are the keypositions. An animation is calculated as a continual change from one key to another
until the last key is reached.

Actions
- Add Key / Add First
With pressing one of these gadgets the active fractal picture in the animationwindow
is defined as a new key and inserted behind the active key or at the first position.
While inserting the program checks, whether this picture is suitable, e.g. whether
it is of the same (fractal-)type and subtype as the ones already in the list, and
whether it differs only in continual changeable parametervalues. If it doesn't suit to
the other ones, then an error-report occurs, in some cases with a hint, why it failed, and
with the offer to adjust the illegal values to the other ones in the list. By the way,
the new key has the same name as the fractal picutre, but it has nothing to do with it. The new key
isn't referenced to the picture, but copied.
That means, that a change to a value of the fractal picture doesn't affect the key with the
same name. This offers a quick method to create an animation. For that purpose
calculate a fractal, insert it as a key, then change a parameter or simply zoom in
and add it as a new key again. Repeat this as long as you wish. Then you only need to set
the desired number of frames of the animation and the animation size, then click on the
start-gadget, then you can watch TV and let the program work...

- Del Key
Makes, what it says...

- Key Up / Key Down
These gadgets alter the position of a key.

- Key to Pic
A disadvantage of copying the keys is, that nobody can change a parameter of a key. But even looking at
the parameters isn't possible. A key isn't a picture, so it can't be calculated,
so parameterwindows can't be opend. If you obtain an animfile from a friend, you are totally
helpless. You can only calculate it, you are not even able to find out, what fractal type this animation
calculates. With the help of this gadget now it's possible to convert a key back to a picture, then

calculate it and change parameters and perhaps delete the old key while inserting the new one.

- Start / Abort
If you click on the gadget 'Start', an animation is launched. For that purpose a fractal window
is opend and a task created in order to calculate the fractal. Of course the program isn't blocked
by the animation. You can calculate as many fractals as you wish, only a second animation you can't
launch. In order to make it clear to the user the gadgets are all disabled, except the
'Abort'-gadget.

- Load/Save
With these gadgets keylists are loaded or saved (and loaded by a friend with a faster
amiga...)

Timesettings
The animation system is now oriented more to the time as before. There exists a
definable timeunit, which can be interpreted as the time which a single frame stays
onto the screen. Now one sees at once, how long an animation will last and at what
time a key will be displayed.

Moment
These two gadgets, both are read-only, show, at what moment the active key will be
displayed. The one shows the time in seconds from the beginning, the other the number
of frames since start of the animation.

relative to the last
The above of the two gadgets, its read only, shows the timedifference to the last key,
i.e. how much time lays between the display of the last key and the active key. The
other of the two gadgets - it can be altered - shows the number of frames, which lies
between the last key and the active key.

total
The two gadgets, both are read-only, show, how long the animation will last, and out
of how many frames it consists. The Duration of course is the number of frames multiplied
with the timeunit...

Timeunit
Here you can set the timeunit. Normally this is the value 0.05, i.e. 0.05 seconds per
frame, i.e. 20 frames per second. If you alter this value, all times will change, but
all frame numbers will be the same. In order to alter the framenumbers, but not the
times, one must click onto 'Normalize Time'.

Normalize Time
This gadget sets the timeunit to 0.05, doesn't alter the moment of a frame, but does recalculate
all framenumbers.
Guess, the timeunit is 0.05 and you would like to increase all framenumbers in order
to 'smooth' the animation. Then you have to set the timeunit to 0.1. This results in
an animation, which lasts twice as long. But it contains the same number of frames
as before. Then you click onto 'Normalize Time'. This now sets the timeunit to 0.05.
It doesn't alter the times, but does recalculate the frame numbers. Because the
animation now lasts twice as long, and the timeunit is the same, twice as many frames are needed.
If you want to decrease the number of frames, act accordingly.

Calculate time
This gadget tries to calculate an ideal number of frames between the last key and
the active key. For this purpose it examines all parameters, in which the two keys
differ and calculates a framenumber according to their differences.

Calculate all
Same as 'Calculate time', but for all keys at once

Others

Buffer
Here you choose the buffer for the calculation of the animation. You need a buffer for 3D-animations or 24 Bit.
I recommend using an IEEESP-Buffer for 3D-animations, because otherwise there may occur some nasty effects due to
the Integer-Buffer.

Interpolation
Here you can choose, whether the single keys should be interpolated linearly, or
whether a spline should be calculated between two keys (currecntly it's a cubic spline).
In case of linear interpolation there occur some jerk-effects, especially when zooming. Well, I personally like this effect...This
will naturally be avoided with the spline-interpolation.

Savemode
Here you can choose, whether the animation should be saved in AnimOpt5-format, so
that the animation can be displayed without problems from any other available anim-player,
or whether every frame should be saved as a single IFF-ILBM-picture. In this case
you can define the basename of every picture right after the start of the animation.
The single pictures then get the basename and a number appended (the framenumber).
This choice is needed, because AnimOpt5 isn't capable of 24Bit-animations. In the case of Savemode=pictures
you can choose the planedepth upto 24 Bit.

Width / Height
- With these gadgets you can define the size of the animation. If you define false values (too big or too small)
then the values are set to legal ones by the program. If 3D isn't chosen, these

values are responsible for the size of the 2D fractal window, otherwise they define
the size of the 3D fractal window.

Planes
- Some fractal types allow calculation of an animtion with a depth upto 8 planes, although
the program runs only o a 6 planes screen or the hardware doesn't allow 8 planes.
Here you can define, how many planes your animation should use. After that you can convert
the animation to your favorite format, HAM6-mode as example.

Startframe&Endframe
These gadgets define the startframe and the endframe. If your computer suddenly
crashes, or if you want to recalculate parts of your animation, you can set these
gadgets to your desired values. If you set 'EndFrame' to 0, then this is the same as
the highest possible framenumber.

3D-Animation
- Some fractal type can be displayed in 3D. If you want this, make sure that this gadget has a checkmark. In this case,
2 windows are opened automatically, the 2D- and the 3D-window and the content of the
3D-window is saved as animkey.

1.12 CycleControl-Window

2.2.5 CycleControl-Window

This window is ment to give you better control over the colorcycling-feature of
ChaosPro.
There exist 3 gadgets:

Colorcycling
- Switches Colorcycling on/off

Speed
- Sets the colorcycling-speed, range from 20 to 999.

Direction
- Should be cycled upwards or downwards?

1.13 User Defined Windows

2.2.6 User defined Windows

You can define an infinite number of windows. All windows have a vertikal button- gadget-bar.
If you click onto a button, then an Arexx-Script is executed. So you can
add some not implemented features to the program.

```
The structure of the windows is defined by the ASCII-file "Windows.asc", which ↩
   must
be in the directory ChaosPro/Prefs
It's structured like following:

WINDOW <windowtitle> <Arexx-Script>
GADGET <gadgetname> <Arexx-Script>
...
GADGET <gadgetname> <Arexx-Script>
WINDOW <windowtitel> <Arexx-Scripts>
GADGET <gadgetname> <Arexx-Script>
...
GADGET <gadgetname> <Arexx-Script>
WINDOW <windowtitel> <Arexx-Scripts>
...
END

The Arexx-Script, that's placed in the line with the WINDOW-keyword, is executed
every time the window is opened. Please make sure that the structure of this file ↩
   is correct, because
otherwise a guru meditation may occur (although I think, this mustn't happen).
If you have wrote this file, then you have to translate it by the
                Preferencesprogram
                .
It creates a file called ChaosPro/Prefs/Windows.prefs.
I hope, the compilation-routine detects all possible errors, so a guru meditation ↩
   is somehow
improbable.
```

1.14 Dockwindows

```
2.2.7 Dockwindows

At this time the program has 3 dockwindows.

Dock1
No. 1 controls all windows, which can be opened separately for every fractal. The ↩
   effect of each gadget
should be clear, if not, you can try it, then you see it...

Dock2
No. 2 controls all windows, which can only be opened once throughout the program
.

Dock3
No. 3 offers some actions, which can be applied to fractals. You find these ↩
   actions in the menu, too, but I think it's easier to
just click onto a gadget than to choose a menu item. The first two gadgets call ↩
   the Undo/Redo-routines of ChaosPro. They are
symbolized by 2 different small fractals, which are (apparently?) in a list. 'Undo ↩
   ' lets you walk back in
the list (arrows point to left), 'Redo' lets you walk forth (arrows point to the ↩
   right side).
```

Below these 2 gadgets there are the actions 'Box zoom in' and 'Box zoom out', below these
a gadget, which executes the routine 'SavePictureToIFF'.

1.15 Formeleditor für Julia/Mandel

2.2.8 Formeleditor für Julia/Mandel

ChaosPro now offers a powerful formula editor for Julia- and Mandelbrotsets, so you can type in your own formula and create a

Julia- or Mandelbrotset
of it.

It's very useful to first read some stuff about Julia- and Mandelbrotsets. Otherwise you will
have serious trouble to understand the following chapter. In fact, I personally
always have some trouble, if I want to create a new formula. It's quite difficult ...
As far as I know, the formula editor of ChaosProV2.0 is the most powerful one on either the
Amiga, the Atari, the Macintosh and the PC, unfortunately it's also the most complicated
one...

Now let me explain the features of it. After that I'll explain the gadgets, if
necessary:

ChaosPro divides every formula in some parts. The first part is the initialization part.
All formula editors, which I know, have only the ability (or don't have even this feature),
to specify one single initializer. But, just as an example, for the calculation of
Mandelbrotset it's necessary, to examine the orbits of all critical points. If you look at the standard
formula z^2+c, then there's only one critical point (nullpoint of the first derivation), and this is the
0. In order to input this formula, you of course would have to specify only one
initializer, the '0'. But other formulas of course can have several critical points, and all of them have to be examined
to get the real Mandelbrotset. Well, ChaosPro is able to do this...

The second part of a formula is the part about the iteration formulas. This part can
consist of several subparts, too. In order to input the standard Mandelbrotset, of course
here again only one part has to be specified, 'z^2+pixel'.
Every iteration element consists of 2 formulas, one formula, which is the condition for the iteration,
and the second formula, the iteration function itself. In order to understand the whole system,
it is necessary to know how ChaosPro calculates the points:
Let us assume, you have a point z, which is initialized to the first critical value.

Then the iteration part starts. At first the condition of the first element is evaluated.
If the result is 'true' (well, not 0...), then the corresponding iteration function is applied to
z. Otherwise the next iteration element is taken, the condition is evaluated, if it is true, then
this iteration function is applied and so on and so forth. If none of the conditions was true, then there's
some sort of a syntax error (this shouldn't occur, otherwise the creator of the formula made a mistake). The program
then just takes the last iteration element and applied it to z. Seems quite strange, doesn't it?
Well, let me explain the reason for this behaviour:
Barnsley, a fractal expert, examined a special class of fractals. He changed the iteration formula while iterating.
For example, if the real part of z is less than 0, then the function (z+1)*c is applied, otherwise the function
(z-1)*c is applied. With the help of ChaosPro you now are able to calculate such fractals,
too.
The third (and the last) part of a formula is the abort part. Here you can specify an abort condition. If this condition is true, then
the iteration procedure is stoped. The 'normal' abort criteria is something like
sqareroot[real(z)*real(z)+imag(z)*imag(z)] > Bailout. If you know something about
biomorphism, then you already know, that it makes sense to change this simple criteria.
Now you can specify a whole formula for this purpose. All the 'normal' abort conditions like 'test for infinity', 'test for
a finite attractor', etc. are still available. Additionally you can choose ' formula defined' in parameter window
number 2, and this 'formula defined' is just the abort criteria you specified with the formula. If you choose it, then
the iteration stops as soon as the condition is 'true'. Now you can define your own areas, perhaps
abort, if real(z) less than -10 or something like this?

Now let me algorithmically write down, what I said:

How does ChaosPro now calculate a fractal, which uses a user defined formula
?

1. Take the first initialization element

2. evaluate the initialization formula, thus initialize 'z', thus z = result of formula
evaluation

3. Take the first iteration element
4. Evaluate the condition of this element
4a. if condition='true', then apply the corresponding iteration function to 'z' and go to step 5.

4b. if condition='false', then take the next iteration element and go to step 4.

4c. if no more iteration elements exist, then apply the last iteration function to 'z'

5. check for all specified abort criteria, check if MaxIt is reached.

6a. if no abort, then go to step 3

6b. abort, take the next initialization element, if one is available, and go to step 2. If no element is there, then the point
is completely calculated.

7. A value was calculated for every initialization element. Determine the minimum of this set. Assign this
minimum to the point.

Description of the gadgets

Mode
This gadget determines the contents of the listview.
There are 3 possibilities:
1. All available formulas are shown by their name.
2. All initializers of the chosen formula are shown.
3. All iteration elements of the chosen formula are shown.

Right beneath there is a gadget, which contains the name of the chosen formula
.

4 gadgets right beneath the listview
These 4 gadgets let you add a new formula, delete the active formula, load a formula or
save the active formula. Kind of easy, isn't it?

Gadgets under the listview

Here is the place to change the formula. If you understood the above explanation, then there shouldn't be any questions.
If you didn't understand the text above, then let me know. I then have to rewrite this
section...

Remark:
The first 6 formulas are built into the program, so they are of course faster than if you
would generate them with the formula editor. So don't be confused, if you can't change
them...

2. The parser and its functions
The parser doesn't know a difference between A, B, C, ... and a, b, c, ... More than 2 parameters aren't allowed.
The wasn't enough place for them in the parameterwindow 1. Parameter 1 is always the
parameter, which is the first in alphabetic order.

The parser knows the following functions:

+ - * / ^ - Addition, Subtraction, Multiplication, Division, Potenz
sin - the Sinefunction
cos - the Cosinefunction

```
tan       - the Tangensfunction
asin      - the Arcussinefunction
acos      - the Arcuscosinefunction
atan      - the Arcustangensfunction
abs       - the Absolutfunction ( Absolut value )
ln        - the natural logarithm
exp       - the Exponentialfunction
log       - the Logarithm to the basis 10
sinh      - the Sine Hyperbolicus
sqrt      - Square Root
tanh      - Tangens Hyperbolicus
cosh      - Cosine Hyperbolicus
cotan     - Cotangens
cotanh    - Cotangens Hyperbolicus
conj      - the conjugiert complex of a number
real      - the real part of a complex number
imag      - the imaginary part of a complex number
acot      - the Arcuscotangens
asinh     - Area Sine Hyperbolicus
acosh     - Area Cosine Hyperbolicus
atanh     - Area Tangens Hyperbolicus
acoth     - Area Cotangens Hyperbolicus
arg       - the argument (phase ?) of a number (the angle between 0 and 2*pi)
e         - a constant: the 'Euler-number'
i         - a constant: the imaginary unit
p         - a constant: Pi - the circlenumber
z         - this parameter is the variable of a formula and
            should be anywhere in the formula...
pixel     - the  complex  number,  which  corresponds to the screen position..
.
12.44     - a number - is treated as a constant
&&        - logical AND
||        - logical OR
!         - logical NOT
==        - logical EQUAL
>=        - greater or equal
<=        - less or equal
!=        - unequal
>         - greater
<         - less
t         - TRUE
f         - FALSE
```

3. The parser and its errors

- "Error in formula detected. Brackets are wrong, I think. Not translated ..."

Here you should examine your brackets. The parser encountered the end of the formula while not all the opened brackets were closed, or it encountered a closing bracket, which doesn't match to any opening bracket.

```
- "Error in formula detected. There’s a character I don’t understand. Not translated
...

Here the parser encountered at the beginning of the translation process an unknown character like
"~" and had stopped immediatly.

- "Error in formula detected. There’s something wrong with the operators or the syntax. Not
translated..."

This error mustn’t occur. If it does, send me a mail with the corresponding formula.

- "Formula too complex. More than two parameters aren’t allowed. Not translated
...

This error message should be clear.
Except z you can only have 2 more parameter, consisting of a letter except e, i, p, z,
...

- "Formula error. Number of operators doesn’t match the number of the operands. Not translated...

This error means that during a test-calculation of the formula there were operands left.
or during this test suddenly no operand were left to perform the operation defined by
the operator. This sounds a bit complicated, so I show you now a few examples, which provocate
this error:
a) a**b
Here now the program tries a test-calculation. There are 2 operators, two multiplications.
Always 2 operands are multiplicated together to form a result. One says, that multiplication
is a dyadic operation. To get a correct result, there have to be 3 oprands, but there are only 2, called
’a’ and ’b’. So here are too many operators compared to the number of operands.
b) b b
Here now is no operator available, but 2 operands. The parser starts a test- calculation, has finished
immediatly and regicnizes that not only one operand is left which it would interpret as the result, but two
operands. So there are now too many operands in the "formula"...
```

1.16 Formula editor for IFS

```
2.2.9 Formula editor for IFS

Again I assume, that you know about the theoretical basis of iterated function systems. Otherwise please read
```

```
the chapter about this topic.

Well, the goal is, to define a system of affine transformations, where every affine transformation
has a probability assinged, which determines, how often this transformation is used.
ChaosPro now is able to calculate the attractor of such an IFS.

Description of the gadgets
The listview of course lists all currently known IFS. Above this listview there is a gadget,
which shows you the name of the currently selected IFS and which lets you change this
name.

Right beneath there are 5 buttons, which do the following:
1. Add a new IFS. After you click onto this button, you are prompted for the number
of affine transformations of the IFS. After the creation of a new IFS the number of
transformations can be changed only indirectly by cloning an IFS.

2. Clone the currently active IFS. Again the user is prompted for the desired number of
affine transformations of the new IFS. The values of the active IFS are intelligently
copied into the newly created IFS, that means, if there are less affine transformations, then
of course not all transformations are copied, and if there are now more affine tranformations,
then all transformations are copied and all the values of the unused transformations are set to
0.

3. Load an IFS from a storage device (well, your harddisk...).

4. Save the IFS to a storage device.

5. Delete the currently selected IFS.

If you have specified too few affine transformations while adding a new IFS, then the only possibility to change this is,
to clone the IFS and now specify more transformations.

Below the 5 buttons there is a slider, which shows the number of the transformation,
which is currently shown. Just move it around to see the other transformations. Below the
slider there is the gadget, which shows the probability, with which the transformation is
chosen.

The above part of the window contains a 3x3-matrix, which is responsible for the rotation and scaling.
If you just want 2D-IFS, then of course only the upper left 2x2-matrix is interesting. Set the third row and
```

the third line to 0. Right the matrix there is a vector mit 3 elements, which is responsible for
the displacement. Again, if you just want to create a 2D-IFS, then only the upper 2 elements are of interest,
the third element should be 0.

One additional feature has been added. You can assign a variable (four variables are available) to an element
of the matrix, the vector or the probability, rather that assigning a fixed number.
This really makes much sense, because the parameter window number 1 contains gadgets for 4
variables 'a', 'b', 'c' and 'd'. If you want to examine, what happens, if one constantly changes
the first component of the vector, then it is desirable to calculate an animation about this.
But the whole animation system doesn't know anything about formulas, it only operates on
fractals and their data structure. It would be quite difficult to change the animation system
to operate on formulas, maily because formulas can change not only in values, but rather in size,
too!
So the animation system can only operate on values, which are displayed in the parameter windows.
And this is the reason for the implementation of variables in the formula editor.
You now simply could specify the variable 'a' as the first element of the vector,
then change 'a' in the parameter window 1, add some key frames and then you could
start the calculation of the animation.

Unfortunately you can only type floating point values into the floating point
gadgets of the formula editor. So you can't simply write 'a' into it. Instead of this
the variables are coded as numbers. Instead of 'a' you simply write '100', instead
of 'b' you write '101', instead of 'c' '102' and instead of 'd' '103'. Of course this
means, that 100, 101, 102 and 103 can't be used as normal numbers. But that should
not be a problem. Just write 100.0001, 101.0001, etc.

1.17 Formula editor for L-Systems

2.2.10 Formula editor for L-Systems

Again I assume, that you know about the theoretical basis of LSystems. Otherwise please read
the chapter about this topic.

You'll find a listview in this window, which shows all formulas of type LSystem. And again, above this listview
there is the gadget, which shows the name of the formula, and which you can change ...

Below the listview there are 5 buttons:
1. 'Add': This adds a new formula of type LSystem to the list. The user is prompted for the number of 'rules', which he wished, should his

formula have. After that the user is prompted for the maximal entry size of a rule . Of course, this determines the
size of the formula on hard disk (if saved) and in memory, so don't set this value artificially high. But keep in mind,
if this value is too small, then it's quite annoying. You just type in a rule, press return, and a requester says, that the rule is
too long...You then have to clone the formula. You then can choose higher values
.

2. 'Clone': This button clones the currently active formula. Again the user is prompted for the number of rules of the formula and
the maximal entry size. The formula, which is to clone, is then copied into the new formula.
If you have made a slight mistake while adding a new formula, that means, you have specified a too small
number of rules or a too small entry size (or a too big one...), then you simple have to clone the formula. Correct these
values, then delete the old formula, if you don't need it any more.

Load, Save, Delete should be quite clear. If not, mail me, ok ? ;-)

The lower part of the window then allows you to define the formula. This firstly is the so called 'axiom',
then the angle, which a '+' or a '-' add or subtract, below this a slider, which chooses the rule, which
is shown below. A rule always has the form <Character>=<String>, don't use spaces
!

Well, lets come to the commands, which ChaosPro understands:
(Just imagine a turtle, which can rotate itself, change the color, move forward and make
such things...)

F: Draws a line into the actual direction using the actual color of the actual length.
f: Same as 'F', but doesn't draw the line (well, 'F' draws, 'f' moves...)
+: Rotates the turtle using the actual angle counter clock-wise. At the beginning this
is the angle specified in the formula editor window.
-: Makes the same as '+', but rotates the turtle clock-wise.
[: Pushes all changeable values - the length of a step, the color, the position, the direction, etc. onto
a stack
]: counter part to '[', restores a state of the turtle from the stack, i.e. the turtle gets a new
position, a new direction, a new color, etc...
|: turtle turns back, it rotates by 180 degree...

Special commands:
'a', 'l' and 'c': These commands affect the angle ('a' for 'Angle'), the length of a step of the turtle ('l') and the color ('c' for
'Color').
Of course, there must be number or something like this after these commands in order to change the values. Otherwise the commands would
be useless. In order to enable a fast processing of a formula, the format of the numbers, which follow the commands,

```
is very strict defined. It would cost a huge amount of time to convert a number ↩
   into a computer
readable form.

So the following rules exist:
- Either a number of the form <xx> follows the command, then this is interpreted ↩
   as <Character>=<xx>
Example: a12 or a03 ==> a=12 or a=3  (you MUST write a03, you MUSTN'T write a3!!
)
- Or: A '+' or '-' and after that a number of the form <xx>, then this is ↩
   interpreted as <Character>=<Character>+<xx>
Example: a+02 or a-13 ==> a=a+3 or a=a-13
- Or: A '*' or '/' and after that a number of the form <x.x>, then this is ↩
   interpreted as <Character>=<Character>*<x.x>
Example: c*1.3 or c*0.1 ==> c=c*1.3 or c=c*0.1

Annotation:
1. The command 'a' doesn't rotate. It just affects the rotation angle used by '+' ↩
   and '-'
The turtle can be rotated only with the commands '+' and '-', which use the ↩
   rotation angle.
2. Instead of 'a','l' and 'c' you may use 'A', 'L' and 'C'.
```

1.18 2D/3D-Fractalwindows

```
                2.3 Fractals

2.3.1 The 2D/3D-Fractalwindows

In the 2D-fractalwindow the 2D-fractal-picture is displayed. It corresponds always
to the actual parameterset, so every time a value is changed, it's calculated ↩
   again.
The 2 dynamic systems are already 3D-fractals, they are always shown in the 2D- ↩
   fractalwindow.

The following actions are possible (except for type=Plasma):

1. Cursor-keys or Joystick in port 2
If you press one of these keys, the fractal-picture is shifted 8 pixels to left/ ↩
   right/ up/
down.

2. Spacebar or fire on joystick in port 2
This zooms in the fractal. If enough memory is available, then a short zoom-in- ↩
   movie is
calculated by scaling the picture.

3. Clicking onto the fractal and moving the mouse around
"Grabs" the fractal and moves it around.

4. Doubleclick onto an interesting detail of the fractal.
This action zooms into the fractal and brings the place, onto which you have ↩
   double-clicked,
at the middle of the window.
```

```
--------

In the 3D-fractalwindow always the to the parameters corresponding 3D-fractal is ↩
    displayed.
This 3D-view is only possible with
                julia-/mandelbrotsets
                .
With
                dynamic systems
                 (that are already 3D-fractals) and with
                bifurcationdiagrams
                 it's not possible to calculate a 3D-view.
(What should be drawn as a 3D-view of the bifurcationdiagrams ?)

--------

All other not supported key-presses are transmitted by the keyboardcontrol- modul
to the datawindow of the fractal and if this window also doesn't understand the ↩
    key,
then the events are transmitted to the parameterwindows. This makes it possible to ↩
     press
the shortcut for increasing the iteration-value in the fractal-window which doesn' ↩
    t understand this and
transmits it to the parameterwindow. I've built in this feature, because there was ↩
     always
the wrong window the active one.
```

1.19 Juliasets: Theory

```
                2.3.2 Julia- and Mandelbrotsets

2.3.2.1 Theory: Juliasets

see also:
                2.3.2.2 Theory: Mandelbrotsets
                In the following I'm refering to the standard formula f(z)=z^2+c.
To create a juliaset, the complex number c is changeable at the beginning, but ↩
    fixed during iteration.
Every point in the window corresponds to a complex number out of the complex ↩
    number-plane. The area of the complex
plane is defined by the area-values in the parameterwindow 1.
The question is now, what happens, if you initialize z with the to the screenpixel ↩
     corresponding complex
number and then applies the formula in an iterativ manner.
So:
z=to the screenpixel corresponding complex number
z1=f(z)=z^2+c
z2=f(f(z))=f(z1)=z1^2+c
z3=f(f(f(z)))=f(z2)=z2^2+c
...

The juliaset consists of all points, which don't lead to an attractiv set of ↩
    points, in other words
```

```
(contrapositiv), all points, which don't belong to the juliaset, are attracted by another point
called attractor, or, to be more general, are attracted by a set of points, a cyclus.
This means, that the juliaset isn't this fantastic colored picture, but the black, booring area.
All the points, which are coloured, are the points, which lead to an attractor, and so they don't belong
to the juliaset.

critical points: f'(z)=0, i.e. 2z=0, i.e. z=0
fixed points: z=f(z), 0.51sqrt(0.25-c) and z=infinite

- Fixed points and Eigenvalues

The interested reader of course has checked it: The juliaset is determined mainly by the
attractiv points, points, which solve the equation z=f(z), the fixed points. It's possible,
that a point z0 is a fixed point, but it's not an attractiv point, i.e. if a point is
very close to z0, then it 'flees' away from z0 under some circumstances. So now
the question is, when is a fixed point z0 attractiv and when not. To decide this,
you only have to calculate f'(z), you have to differenciate f(z). In this case
f'(z)=2*z. If you then take a z very close to a fixed point z0 (really very close, infinite close to z0),
and calculate f(z), then you recognize, that approximaly the following is right:
|f(z)-z0|=|f'(z0)*(z-z0)| (compare it with the equation of the tangente in the point z0)
So you now recongnize, that attraction or repulsion depends on the absolut value of f'(z0), the so called 'Eigenwert'
of the fixed point z0. If |f'(z0)|<1, then the distance of f(z) and z0 is shorter than
the distance of z and z0, so z0 is attractiv. If |f'(z0)|>1, then the distance is greater, z0
is repulsiv. If |f'(z0)|=1, then the fixed point is neutral. In this case many other interesting
things may occur.
In the standard formula f(z)=z^2+c you get the fixed points by solving the equation
f(z)=z, i.e. z^2+c=z.
The 2 results are:
z1=0.5+sqrt(0.25-c)
z2=0.5-sqrt(0.25-c)

So if you want to calculate an interesting juliaset, you have to choose 'c', so that the Eigenwerte
of the fixed points are less than 1.

Due to theoretical reasons the infinite point has to be considered as an attractive fixed point, although
it's clear, that in practice this 'point' isn't attractive. To calculate the Eigenwert of infinite
isn't much intelligent. It's always an attractive 'fixed point'. Due to this, most of the other fractal
creation programs only check a point, whether it is attracted by the infinite point, guessing, that it's
```

the only one. Well, this produced nice pictures and that was all they wanted.

All the above mentioned can be made a bit more complicated. At the beginning I mentioned
that not only points can be attractive, but also a set of points, a cyclus with a specific
lenght. This is as example a set of 3 points (cyclus lenght is equal to 3) z1, z2, z3 which fulfil
the following:
f(z1)=z2 f(z2)=z3 f(z3)=z1
altogether: f(f(f(z1)))=z1
If you use f with z1, z2 or z3 three times, then again the result is z1, z2, or z3.
With this program you can also check and search for a cyclus. But don't expect me telling
you the theory, because not even I do understand it...

A bit more about juliasets you can find in the following chapters, where the various
parameters are discribed. There hopefully you realize, how this program works in order to
calculate the pictures.

If you haven't understood this chapter, then the probability is relativ high that
you sit in front of the monitor looking at a black area and wondering why this isn't
a nice, good looking, colorful fractal.

Well, and now something interesting especially for the mathmaticians:
Newton's Way for Estimating Nullpoints

Newton (who other could it be ?) was engaged in getting an approximate value for the nullpoints
of polynomials P(z). The formula he found was:
f(z)=z-P(z)/P'(z)
At the beginning you initialize z with a value, then you apply the formula in an iterativ manner to z again and again
until the functionvalue doesn't change any more. The problem which Newton hadn't solved
and which even now isn't solved is, what initial value you have to choose for z so that
this method converges, and can you in any way get all nullpoints of the polynomial with this
method?

This problem isn't solved at this time. To get a picture, they use the computer.
If you check the method out you will find that it's an ordinery juliaset just with a user
defined formula.
Let be P(z)=z^3+2, so P'(z) equals to 3*z^2
As f(z) you have to define z-(z^3+2)/(3*z^2). Then you have to compute the juliaset of this.
Please note, that nowhere in this formula is a variable parameter c, so it's totally booring, to calculate
the mandelbrotset of this. To get the desired picture, you now have to say to the computer that

```
it should look for finite attractors (these are the nullpoints of the polynom P(z) ↩
    ). So you have to
click on the corresponding gadget in the parameterwindow 2. The resulting image is ↩
     the so called
'Basin of Attraction' of the polynom. If you then open the datawindow, you will  ↩
    find
the endpoint of the calculation beneath 'End:'. This is the finite attractor, a  ↩
    good approximation
for a nullpoint of the polynom. If you move the mouse pointer around, you can see  ↩
    very clearly, what
initial value for z results in what nullpoint.
We now have a polynom of degree 3. There are only 2 cases possible: Either are all ↩
     3 nullpoints real, or
there is a real nullpoint and 2 complex which are conjunct complex to themselves.
The second possibility you can see in the image as you can really easy check out.
If you now look at the fractal you are recognizing quickly why the mathematicians  ↩
    have so big trouble with so
easy to express problems.

Hint:
The tranformation of this fractal into the 3th dimension isn't very attractiv.  ↩
    That's because for finite attractors
the continuous potential method in my implementation fails.
```

1.20 Mandelbrotsets: Theory

```
                2.3.2.2 Theory: Mandelbrotsets

See also:
                2.3.2.1 Theory: Juliasets
                Mainly there are 2 different types of juliasets:
Type A) The juliaset is 'dusty', i.e. it consists of infinite many incoherent  ↩
    points.
Type B) The juliaset is 'coherent', i.e. it consists of a variety of lines, areas, ↩
     or something like this.
The type of a juliaset is determined by the parameter, in the case of the standard ↩
     formula
f(z)=z^2+c it is determined by 'c'.
The mandelbrotset now shows graphically, for what values of 'c' the corresponding  ↩
    juliaset is
'dusty' or 'coherent'.
Julia, the inventor of the juliasets, has thought out a trick to decide, wheater  ↩
    the juliaset is
dusty or coherent without calculating the whole image. To do this, only the  ↩
    critical points are to examine.
The critical points of a formula are the points, for which f' is equal to 0. If f( ↩
    z)=z^2+c, then
f'(z)=2*z, so the only critical point is 0. To build the mandelbrotset all  ↩
    critical points have to be examined.
This version of the program can't do this. It can only examine one critical point  ↩
    at a time. Due to this, the resulting image
isn't correct, if more than one critical point is existant. To get the right image ↩
    , you can go the following way:
```

```
Let the program examine the critical points one after another and save the images ↩
    to disk.
If you have done so, then start a paintprogram and paste the pictures to one ↩
    picture together.
The result is the correct mandelbrotset.

So a mandelbrotset is made like follows:
Dependent on the area out of the complex plane you initialize c with the to the ↩
    screenpixel corresponding
complex value and z with the critical point. Then you iterate the formula, i.e. ↩
    you calculate f(z), then f(f(z)), etc.
If the result leads to infinite, then the juliaset determined by the value of 'c' ↩
    is dusty. If it leads
to a finite attractor - a point or a cyclus - then the coresponding juliaset is
coherent.

Due to this the mandelbrotset is a kind of a landscape for all the juliasets you ↩
    can
calculate from a formula. Very often it happens that you have big troubles setting ↩
     a
suitable value for 'c' to calculate a nice juliaset. In this case you only have to
calculate the corresponding mandelbrotset. In this image you have to look where ↩
    your
booring parametervalue lies and so you have the answer to the question, why the ↩
    image
isn't nice. A solution now is, to choose a value for 'c' which lies at the border ↩
    of the mandelbrotset.
There you can expect that the corresponding juliaset doesn't know exactly, if it ↩
    is dusty or coherent
(of course, the juliaset knows it, but not the computer). Inside the mandelbrotset ↩
     the
juliaset is mostly a big ugly area. Outside it's dusty, almost all points don't ↩
    belong
to the juliaset, but lead very quickly to an attractor.

For a very simple choose of the parameter of a juliaset see the menuitem Set ↩
    Juliaparameter
```

1.21 2.3 Fractals --- 2.3.2 Julia- and Mandelbrotsets

```
                2.3.2.3 Parameterwindow 1

Parameter
- Dependent on the formula there are 0, 1 or 2 complex parameters choosable. These ↩
     are to define here.
If 2 parameter are to be chosen, then the first parameter matches the parameter ↩
    which comes
first in the alphabet. The parameter is for juliasets decisive because these ↩
    define the exact locations
of the fixed points and the Eigenwerte. They are to be defined in respect to this.
For a simplified choose of the parameter it would be very comfortable, if it would ↩
     be shown
in the mandelbrotset. This can be done by choosing the menuitem Set Juliaparameter ↩
    . The
```

mandelbrotset of z^2+c as example is a kind of a landscape for all juliasets z^2+c ↩
.
Interesting juliasets can be found at the border of the mandelbrotset.

The first parameter can’t be chosen in conjunction with mandelbrotsets because it’ ↩
s
initialized with the to the screenpixel corresponding complex value.

Iterationen
- The quality of a julia-/mandelbrotset depends widely on the iterationvalue. The
higher the better, but also the slower the calculation. With the slider you can
easily change the value without the keyboard. If you click onto it, then it adds ↩
its value
to the actual iterationvalue. If you then let it, it snaps back to the ↩
nullposition.
Alternatively you can make a greater change of the iterationvalue by directly ↩
inserting the
new value into the gadget. After that you must leave the gadget by pressing the ↩
return- or the tab-key.
As described in the theoretical chapters, all points must be filtered out, if they
are of an ordered type, i.e. if they are attracted by a set of points (juliaset) ↩
or
if they are attracted by the infinite attractor (mandelbrotset).
Due to the optical appearance these ’ordered’ points are colored according to the ↩
number
of iterations it lasted, until the program recognized, that the point is ’ordered ↩
’.
All these points belong to the outside of the julia-/mandelbrotset. If a point
after the adjusted number of iterations isn’t of an ’ordered type’, then it is ↩
considered to be of ’chaotic type’
and drawn as if it belongs to the julia-/mandelbrotset.

Passes
- With this you can set the number of draw-passes. Due to the condition of the ↩
julia-/mandelbrotset
one can draw a conclusion out of same iterationvalues at the corner of a rectangle ↩
.
Because there are ’bands’ of same iterationvalues, the whole inner of the ↩
rectangle most probably
is of the same iterationvalue. Well, this conclusion isn’t always correct, it’s of ↩
course
totally wrong for ’dusty’ juliasets, but it helps to decrease the calculation time ↩
.
And anyway, if you have a ’dusty’ juliaset, you don’t see the dust even if you ↩
choose ’1-Pass’
which means that no conclusions are made, every pixel is calculated.
This is because it’s totally improbable that out of the limited number of points ↩
which are
drawn even one single point falls into the dusty juliaset. They fall almost always ↩
a very very little
beneath the juliaset and so the points are considered to be of ’ordered type’ and ↩
don’t belong to the
julia-/mandelbrotset. To avoid this, other calculation methods would have to be ↩
implemented, the
distance-method with continuous potential as example, which calculates the ↩
distance of every point from
the juliaset (yes, this is possible).

Ausschnitt
- The juliaset shows what happens with the points of the complex plane, if you apply
the formula in an iterativ manner to every point of the plane. Here you can choose the
area out of the complex plane.

- If you draw the mandelbrotset, this defines the area out of the complex plane which is
to be used for the parametervalue 'c' of the formula.

Eliminate
If this gadget has a checkmark, then the program searches after every drawing pass for areas,
whose corners have all the same iteration value. If it finds such areas, it assumes, that all
points inside this area are of the same value and doesn't calculate them. This saves quite a
bit time (about 30% on average). Of course this asumption about the inner values of areas isn't
totally correct, but you normally won't notice it.

Angle
The 2D-fractal is accordingly rotated. This is interesting especially in conjunction
with the animation system. There exist 2 gadgets to control the rotation angle. The range
goes from -30000 to 30000 (degrees...).

A little theory to:

2.3.2.1 Theory: Juliasets

2.3.2.2 Theory: Mandelbrotsets

1.22 2.3 Fractals --- 2.3.2 Julia- and Mandelbrotsets

2.3.2.4 Parameterwindow 2

Outside coloring
- several possibilities are offered:
1. Color
With this the whole outside area is drawn with the below setted color. Because the outside area
normally is responsible for the nice appearance of the fractal, the sense of this choose
is't clear at the first glance. But with this one can better display the dusty appearance
of the juliaset, if it's possible. Now the many colors don't disturb.

2. Iteration
Now every pixel of 'ordered type', i.e. which is attracted by an attractor, is colored

according to the number of iterations is lasted until it was clear that it is attracted.

3. CPM - an Acronym for Continous Potential Method
Whoever had a look at the second point, recognized that obviously only an integral number
can be attached to every point. That's especially bad while calculating 3D views of fractals
because here 'stairs' appear in the view. The picture looks like a terraced landscape.
The heights jump from one value to another. This changes, if you choose this method.
By means of a fairly simple function the outside area of a julia- or mandelbrotset can be
transformed to the inside area of a circle with radius 1. The fascinating thing is now that
the boundaries of the coloured bands, these extremely complicated curves, are transformed in
concentric circles with the middlepoint 0. These circles now have a radius of course and
now you can replace the iteration values with these radiuses. Another big advantage
is that a point which lies in the middle of a single coloured band is transformed into the
circle between the two concentric circles defined by the boundaries of the neighboured
iteration-bands. If you now define a circle with the middlepoint 0 which contains the one point
and revert the function, you'll obtain a new boundary of a band with a very complicated structure
but which lies perfectly between the boundaries of the old bands!
This method now is used to attach a real (not integral) value to every point so that
this terraced effect in the 3D-view is avoided. With 'Mult.' you now can determine, with what number
this real number is multiplicated. The program remembers no real numbers, but only integral numbers, because
these don't need so much memory. So it stores integral numbers, the old real numbers multiplicated with 'Mult'.
This means now that a value of 100 for 'Mult' allows the program to calculate 100 values between
two iterations. This is really enough for avoiding the terraced effect. If you then save
it as a 24-bit fractal, the 100 additional values between two iterations are used to
calculate additional colours between 2 bands.

4. DEM - an acronym for Distance Estimator Method
This algorithm basically is an enhancement of CPM. This method calculates for every point an estimation
for the distance from the point to the border of the J-set or M-set. Of course this is a real number, not an integer,
so this mode is great for saving in 24 bit and transforming the whole fractal into 3D.

5. DEM Border

This is just the same as DEM, but the 'Bailin'-value is used in a strange way. Sometimes one wishes to just draw the edge/border of the M-set or the J-set. That means, one just wants to calculate all points, which have a distance of less than 'l' to the edge of the fractal . And this 'l' you can specify with 'Bailin'.

Annotation: In former times (ChaosProV1.0) it was a big problem to transform some Mandelbrot- or Juliasets into 3D, so they look good. One choosed CPM, but many sudden jumps occured, so the whole fractal was totally screwed up. DEM now is better in many of these cases. 2 points, which are tight to each other, could get very different values, if you use CPM, but if you use DEM, they get similar values. So if you weren't able to find values, so that the 3D- transformation of a fractal looks good, then try DEM.

Inside coloring
- Now there are 6 possibilities:
Color - Infimum - Infimumsindex - Supremum - Supremumsindex - Magnitude of z
The inside area normally is one-coloured. But this can be avoided. There are some methods for assigning colours to points of the inside-area.
Let be (z, z1, z2, z3, z4, ..., zn) the way of a point z, where n is the maximum of iterations.

Infimum
Here the infimum (well, it's of course the minimum, it should be the infimum) of a point is calculated, i.e. the minimum of |z1-z|, |z2-z|, |z3-z|, ..., |zn-z|. The minimum is multiplicated with 'Multiplicator' and stored as an integral number.

Infimumsindex
Here the index of the infimum is calculated, i.e. the number of iterations, when the infimum (minimum) appeared. If the minimum of |z1-z|, |z2-z|, |z3-z|, ..., |zn-z| is equal to |z3-z|, then the index is 3.

Supremum
Here now the supremum (in reality it's the maximum) of a point from the start- value is calculated, i.e. the maximum of |z1-z|, |z2-z|, |z3-z|, ..., |zn-z|. The result is then multiplicated with 'Multiplicator' and stored as an integral number.

Supremumsindex
Here the index of the supremum is calculated, i.e. the number of iterations, when the supremum (maximum) appeared.

Magnitude of z

After the maximum of iterations the absolut value of z is calculated, multiplicated with
'Multiplicator' and stored.

I was inspired to implement these methods by the book 'The Beauty of Fractals', page 62,
and of course by FractInt from the IBM-PC-Clones

Abort conditions
Here you can define, what kinds of attractors should be able to stop the iteration sequence.

1. infinite
If checked, every point is examined, whether it is attracted by the infinite point.

2. finite
If checked, every point is examined, whether it is attracted by a single finite point.

3. cyclus search
If checked, every point is examined, whether it is attracted by a cyclus (a set of points,
may be one single point). For this to work, you have to define the field 'Start'. There
you set the iteration-value, from when the search starts. It should be about the half of the
maximum number of iterations defined in the parameterwindow 1. This is needed, because a point
goes round relativ randomly on its way until it decides to be attracted by a cyclus.
So you should give a chance to the point to take a decision.

4. Formula defined
If choosed, then in every iteration cycle the abort condition, which is specified inside the formula,
is checked, too. Of course, this only works with user defined formulae and not with
the built in formulae.

Bailout
If every point is examined, whether it is attracted by the infinite point, then the question is,
how to determine, whether a point is attracted by the infinite. The following method
is applied almost everywhere (for exceptions see {"Biomorphy" LINK ExpertJM_Bio})
You define a circle with the middlepoint 0 and the radius 'Bailout'. If a point in its way falls outside this circle,
then its considered to be attracted by the infinite.

Bailin
If every point is examined, whether it is attracted by a finite point or a cyclus,
then the program defines a circle round this finite point with the radius 'Bailin'.
If a point falls on its way inside this circle, then the point is considered to be attracted by
the corresponsing point.

Theory:

2.3.2.1 Theory: Juliasets

2.3.2.2 Theory: Mandelbrotsets

1.23 2.3 Fraktale --- 2.3.2 Julia- und Mandelbrotmengen

2.3.2.5 Parameterwindow 3

Circle inversion
- This is a geometrical transformation. It throws all outside the circle defined by
'Middlepoint' and 'Radius' inside the circle and vice versa.

Biomorphy
- Normally Bailout and Bailin define circles.
Whenever points fall outside or inside the circles, the iteration sequence is stopped.
But why should one define circles? Somebody experimented and defined rectangles and other
areas and tested accordingly. The results are fractals which look a bit like a microorganism.
That's why its called 'biomorphy'.
The exact abort conditions (the areas) are defined like follows:
(x is the real part of z, y the imaginary part)

for bailout:
abs(x)+d*abs(y)>Bailout
and/or
d*abs(x)+abs(y)>Bailout

for bailin:
abs(x)+d*abs(y)<Bailin
and/or
d*abs(x)+abs(y)<Bailin

Whether to connect these two inequations with 'and' or 'or', you can define by the cycle-gadget. In the program the variable
'd' is called 'Biomorphyvariable'.

If you set d=0, then:
abs(x)<Bailin
and/or
abs(y)<Bailin

In the case of 'and' a rectangle, in the case of 'or' a cross.

Decomposition
- Now the outside area is subdivided in fields of angles. You define the number of fields
by the value in 'Coding'. Every end-value of a point (i.e. after the maximum number of iterations)

```
is examined, in what field it lies and coloured accordingly.

Virtual fractal calculation
Starting with V2.0 ChaosPro has the ability to calculate fractals in a virtual  ↩
    mode.
This means, it's possible to calculate huge fractals without the need for dozends  ↩
    of megabytes
of RAM. For example, up to now it was impossible for me, to calculate a 3D-fractal ↩
     of size 1024x768 in 24 bit, because I don't
have enough memory for the required buffer.
To calculate a fractal in virtual mode, you have to specify the width, the height, ↩
     the depth in planes, and whether a 3D-transformation
has to be performed after calculating the fractal. If you have specified these  ↩
    values, you have
to choose the menuitem 'Fractal/Start virtual'. ChaosPro will then start the  ↩
    calculation process. The buffer now
will not be created in RAM, but on your storage device. A file will be created on  ↩
    this device.
Default directory is 'ChaosPro:', but this can be altered with the tooltype ' ↩
    Virtual=<Dir>'.
As soon as the fractal is finished a filerequester appears, asking you for the  ↩
    name of the IFF-ILBM-file to store
the big fractal in it.
This virtual calculation is useful especially in conjunction with the printer tool ↩
     'Studio',
which allows you to print huge fractals, which really look excellent. Normally a
fractal of common size will somehow consist of many little squares, which doesn't  ↩
    look
very good. Now you can print posters...

Theory:
```


1.24 2.3 Fraktale --- 2.3.2 Julia- und Mandelbrotmengen


```
With this window the most important data of the julia-/mandelbrotset can be  ↩
    examined.
The window takes the mouse-position and calculates the exact values at the  ↩
    corresponding place.
- The field 'cause' shows, why the calculation ended.
- The field 'iterations' shows, when this was the case.
- The field 'distance' shows, how far away the point is from the edge of the set
.
- 'Point' and 'Start' contain the same values in conjunction with juliasets. In  ↩
    the field 'Point'
the complex number at the mouse-position (Pixel x and Pixel y) is shown. In the  ↩
    field 'Start'
the initialization value of z is shown. In conjunction with Juliasets this of  ↩
    course is the same as
```

in 'Point'. But in the case of mandelbrotsets this is the critical value, which can depend on a formula, which must
first be evaluated, like with formula 2 ('m/(2m-2)').
- Infimum and Supremum
Here the minimum and the maximum of the distance of the orbit of the point to the origin is shown. Additionally the index is shown, that means, in what iteration cycle this
happened. For a deeper understanding of what I'm talking about let me refer to the coloring modes for the inside area Infimum/ Supremum/ Inf.index/
Sup.index.

Theory:

2.3.2.1 Theory: Juliasets

2.3.2.2 Theory: Mandelbrotsets

1.25 2.3 Fractals --- 2.3.2 Julia- and Mandelbrotsets

2.3.2.7 The Formula window

ChaosPro has a relatively good editor for creating user defined formula. These formulas you can use in conjunction
with
Julia- /Mandelbrotsets
.

This window lets you specify the formula to use for the Julia- or the Mandelbrotset. Please notice, that
all the different coloring modes like CPM and DEM are only available for the first formula in the listview. These modes can't be
selected, if you use another formula.

1.26 2.3 Fractals --- 2.3.2 Julia- and Mandelbrotsets

2.3.2.8 Colormapping Window

This window lets you have good control over how the colors are distributed over the various iteration
values. ChaosPro uses builtin functions, which are applied to the iteration values . The result
then determines the color.

In order to determine the function there exist the gadgets Type, Factor, Max and Min.

'Min' determines the minimal iteration value. If a point has a value less than ' Min', then in all cases
it gets the color 4.
'Max' and 'Factor' have an effect on each other. The whole range starting from ' Min'

upto 'Max' is assigned the whole palette, i.e. if there are just the iteration values
from 'Min' to 'Max' in ascending order, then there will appear the palette itself.
If you change 'Max', then 'Factor' will be adjusted automatically and vice versa.
'Factor' is the value, which the program directly uses, whereas 'Max' is more intuitiv for
the user.

Well, let 'ItValue' be the value, which the program has calculated at a specific
position (iteration value), NumCol is the number of available colors for drawing the fractal,
i.e. on a 8 bit display 256-4=252 (the first 4 colors aren't used...).

The following functions are available:
Linear
The function looks like this:
Color=(ItValue-Min)*Factor/100 Modulo NumCol + 4

This function is the most primitive (and worst) function. Normally it is used by almost every
other fractal program, because it's so easy...

Sin
Color=abs(sin((ItValue-Min)*Factor/10000))*(NumCol-1)+4
This function creates some interesting effects when you switch on the colorcycling
mode. Due to the constant walk through the whole palette back and forth this function is
good for most of the fractals.

0.75
Color=((ItValue-Min)*Factor/100)^0.75 Modulo NumCol + 4

Log
Color=log((ItValue-Min)*Factor/100+1)*NumCol/6 Modulo NumCol +4
Good, if you have zoomed in many times and there appear too much too different
values...

ArcTan{ub}
Color=abs(arctan((ItValue-Min)*Factor/10000))*2/PI*(NumCol-1)+4
Quite good, because in every case higher iteration values will result in equal or higher
color values. The palette is used only once, so this function is very good for
deep zooms.

Sqrt
Color=sqrt((ItValue-Min)*Factor/100)*Factor/20 Modulo NumCol + 4

In order to get a good impression of how the function with the parameters looks like, there exists
a graphical representation of the function.

If you click onto the gadget 'Suggest', then the program searches the whole buffer (if available) and
sets 'Min' and 'Max' to the minimal and maximal values appearing in the buffer, thus
forcing, that the palette will be used only once.

1.27 2.3 Fractals --- 2.3.3 Bifurcationdiagrams

2.3.3 Bifurcationsdiagrams

2.3.3.1 Theory

I'm explaining the theory on the Verhulst-Model f(x)=a*x*(1-x)

This model can be interpreted as follows:
Let x be the population of a race, e.g. of hares. It's normalized, so that x s ↩
from the range 0 to 1.
0 means, there is no hare, 1 means, the whole natur is full of hares, no more ↩
hares are
in any way possible. Then let a be the growth rate of the population. a=1 would ↩
mean that
the population of the hares doesn't grow. So only the factor (1-x) is to explain.
It's a measure for the free place in the nature, which remains to the hares and it ↩
can be interpreted as
the available amount of food, which lies again between 0 and 1.
From one year to the next the population now is calculated by simply applying the ↩
function f(x)
to the population x.
Now lets have a look at this model:
Let a be equal to 2 (this is a really reasonable value):
If in one year the population is little, then there's much food available, so the ↩
population will grow.
If the population is big, then there's less food left, so the hares die by ↩
starvation.
The question now is: What balance of the population will be the result in many
years?

Let be x0=0.1, a=2, then in the following years the population is:
x1=2*0.1*0.9=0.18
x2=2*0.18*0.82=0.2952
x3=2*0.2952*0.7048=0.416
x4=2*0.416*0.584=0.486
x5=2*0.486*0.514=0.50
...
Here the balance is reached quickly at 0.5 and this is the result, that means, the ↩
population of the hares
would be grow upto 0.5 and then constantly be at this value.
But what happens if you alter the growth rate? The question is, what balance is ↩
reached, anyway, is a balance reached?

This model is a very simple one, but there are many surprising effects already in
it.
1. case: 0<a<=1

In this case x converges to 0, and this is clear, because a is our growth rate, so
the hares don't have enough children, they'll die.

2. case: 1<a<=2

Here now the population reaches quickly a balance situation, the population is ↩
growing or shrinking in a monoton
way to the balance, depending on the startvalue of x.

```
3. case: 2<a<=3

Here also there's a balance, but the successive values of x converges in an ↩
    oscillating way
to the balance-point and not in a monoton way.

Now let a be grater than 3 z.B. a=3.1
x1=0.3
x2=0.651
x3=0.704
x4=0.646
x5=...

If you calculate further, then you'll recognize, that x oscillates between two ↩
    values,
0.557 and 0.764. So here we don't have a balance, the population of our hares is ↩
    springing
from one year to another between the two values.

If you then take a grater a, but less than 3.449489, then always the population ↩
    oscillates
between 2 values. But then something happens again: a period-doubling, that means, ↩
     a oscillates
between 4 (!) values. At a=3.5441 this 4-cyclus changes to an 8-cyclus. All these
values, at which the cyclus lenght doubles, are called bifurcationnodes.
This 8-cyclus mutates to a 16-cyclus, then to a 32-cyclus, etc., upto a specific ↩
    value:
a=3.569946
From this value upto a=4 there it happens:
The whole thing gets chaotic, that means x oscillates randomly between any values,
here now the attractor isn't a cyclus with a fixed length, but a one dimensional
fractal. In this area upto 4 there are a few 'windows', e.g. at a=3.83, where
a cyclus with the lenght 3 dominates, which mutates to a 6-cyclus, then to a 12- ↩
    cyclus, a 24-cyclus, etc.
Windows like this are all over this area upto 4.

If you now look again at this model and remember, how we have started, then you ↩
    surely
are surprised, what strange things can happen in such a simple model. At the first ↩
     glance
you surely had thought, that there simply have to be any balance...
```

1.28 2.3 Fraktale --- 2.3.3 Bifurkationsdiagramme

```
                2.3.3.2 Parameterwindow 1

Formel
- In the previous chapter the Verhulst-formula was examined. But other formulas ↩
    may also be used.
In this program 5 of the more important formulas were built it. You can draw the ↩
    bifurcation-diagram of these.

Iteration
- In order to draw the bifurcation diagram correctly, the initial value has to be
```

```
iterated sufficiently often to give it a chance to be attracted by a probably ↩
    existing attractor.
Only after this the program can draw the diagram correctly. In case of the ↩
    bifurcation diagram the initial value is iterated half of the value, which is ↩
    here
defined. Then the initial value hopefully has reached its attracting cyclus. Then
the point is iterated further, until the value in this field is reached, but now ↩
    the various results are drawn.
If you want to draw the diagram more exactly (perhaps if you have zoomed into a ↩
    bifurcation), then
you'll recognize, that it's not a sudden occurring bifurcation, but a wide band of ↩
     various points.
This isn't correct. It's the program, which isn't exact enough. There are really ↩
    suddenly occurring bifurcations.
In this case you should increase the iteration-value, the program then is more ↩
    exact while calculating. And then
there are again real bifurcations (until you don't zoom in further ...)

Variable x/Variable y/both
- This option is only available with formula 3. There you have the formulas
x=a*x*(1-x-y) and y=a*x*y, so there are 2 variables, perhaps the foxes and the ↩
    hares (and the growthrate)
Which variable the program should draw, you determine with that.

a: Minimum - Maximum
- In the fractalwindow  horizontally the parameter a is drawn. Here you define the ↩
     minimum and the maximum value of a (the growthrate).

x/y: Minimum - Maximum
- In the fractalwindow vertically the variable x - in conjunction with formula 3 ↩
    also y -
is drawn. Again here you define the minimum and the maximum.

Theory:

                Chapter 2.3.3.1
```

1.29 2.3 Fractals --- 2.3.3 Bifurcationdiagrams

```
                2.3.3.3 Datawindow

- In the fields a and x/y the values corresponding to the actual mouse position ↩
    are shown.
In the fields x and y and in the fields End x and End y the start values ( ↩
    initialization values) and the results (endvalues)
after the here below defined number of iterations are shown.

- In the gadget cyclus the lenght of a eventually found cyclus is shown.

- Through the field 'Show iteration' in conjunction with the slider is defined, ↩
    after how many
iterations the values of x and y are transferred into the two fields 'End x' and ' ↩
    End y'. This enables examining
the various values of x (and y) without using the calculator.
```

Hint:
Though pressing the key 'I' or 'Shift+I' this value can be changed from the fractalwindow.
So it's not necessary to activate the datawindow.

Theory:

Chapter 2.3.3.1

1.30 2.3 Fractals --- 2.3.4 Dynamic Systems

2.3.4 Dynamic Systems

2.3.4.1 Theory

In the year 1961 the moteorologist Edward Lorenz examined a system of a few differential equations, a system, of which not the concret points are known, from which one can calculate another, but the derivation of every point, so that an approximation of the next point can be calculated.
Well, he made his experiments and found out, that his result depends very strongly on the used numerical precision. A very small error at the beginning caused a totally different result. So the title of one of his publications was: "Kann das Flattern eines Schmetterlings in Brasilien einen Orkan in Texas verursachen?" (Can the fluttering of a butterfly in Brasil cause a hurricane in Texas?"). The answer was yes. So this by Lorenz discovered effect is called "Schmetterlingseffekt" (in english perhaps "effect of a bufferfly"). Edward Lorenz then simplified his model and experimented with it. It contained only 3 differential equations:

dx/dt=-ax+ay
dy/dt=cx-y-xz
dz/dt=-bz+xy

They are read like follows:
The derivation in x-direction to the time is -ax+ay
The derivation in y-direction to the time is cx-y-xz
The derivation in z-direction to the time is -bz+xy

Lorenz gave a the value 10, b the value 8/3 and experimented with different values of c. The resulting object can be interpreted as a 3-dimensional curve and, if an initial point is given, it's from a mathmatical point of view (theoretically) definite, but not in practice.

Lorenz took the value 28 for c and calculated the curve for various initial points.
But although the curve started totally different, he found, that after a few seconds always

```
the same figure appeared. It had a very complicated structure, it was built from  ↩
    an infinite number
of loops, and the whole thing was very strange... It looked like the 3  ↩
    differential equations
"stamped" a very complicated structure into the 3-dimensional world in his  ↩
    computer, a fractal attractor,
which attracts every point in this world. Because the structure was so complicated ↩
    , such an attractor
is called "strange attractor".

Well, some of the numericans call all these effects as totally feeble-mindedness,  ↩
    because
it's all caused by rounding-errors from the computer, so the whole story exists  ↩
    only
in the computer and has no practical meaning. But I don't think this is correct.
Fact is, that also the real nature only calculates with integer-values. That means ↩
    , nothing
in the nature is infinitely often dividable, from all there is a smallest thing.
The lightquant, the quarks, etc. That means now, that also in the nature there  ↩
    must occur
rounding-errors. And so the computers are perfect imitators of the nature, at  ↩
    least qualitatively, not
quantitatively, because the number of integer-numbers in the nature is a bit  ↩
    greater than the
corresponding in the computer world.
```

1.31 2.3 Fractals --- 2.3.4 Dynamic Systems

2.3.4.2 Parameterwindow 1

```
Area
- Because the Lorenz/Roessler-attractor is threedimensional, it's a little problem ↩
     to define the
drawarea. In this program this is solved like follows: You define the values as if ↩
     you are looking
at the attractor from the front. So you define the drawarea.

Viewangles
- In order to not only view the attractor from the front, but from any point in  ↩
    the room, you can
change the viewangles. The system, which is here used, corresponds to the system  ↩
    of the earth: a degree of latitude and
a degree of longitude. With alpha you define the degree of longitude, with beta  ↩
    the degree of
latitude.

Parameter
- Here you can set the 3 parameter used in conjunction with the dynamic systems.
I recommend to change the values only slightly, because the systems react heavily  ↩
    to little
changes.

Systemtype
- At this time the program offers 2 types out of the class of the continual  ↩
    dynamic systems,
```

the Lorenzattractor and the Roesslerattractor. The Lorenzattractor is defined by
the following 3 differential equations:

dx/dt=-ax+ay
dy/dt=cx-y-xz
dz/dt=-bz+xy

The Roessler-attractor by these:

dx/dt=-y-z
dy/dt=x+ay
dz/dt=b+xz-cz

Theory:

Chapter 2.3.4.1

1.32 2.3 Fractals --- 2.3.4 Dynamic Systems

2.3.4.3 Parameterwindow 2

Startpoint
- In the theoretical chapter there was mentioned, that independently from the startpoint the way
of the point is always attracted by an object, which is called "strange attractor" due to its complicated structure.
Everybody, who doesn't believe this, has now the possibility to check this out and to change the startpoint.

Time settings
- With 'Time' the duration is defined, how long the point is drawed. The differential equations system
describes the change of the way in dependence of the time. But because the computer can't do anything with
a derivation, it must replace this 'dx/dt', with is equivalent to the limes of delta x divided trough delta t
for t to null, by delta x divided trough delta t with an adequately small delta t. This is what you
can also choose.

Drawing speed
- The Lorenz- and the Roesslerattractor can be drawn really quick. That's nice of course. But
to examine the structure, to see, how it's made, it's much too quick. So if you want to see the
attractor being built, then you must slow down the drawing speed. With this slider you
can set the speed from 1 (slow) upto 100 (as fast as possible).

Drawmode
- There are 3 possibilities offered:
draw as points/draw as Lines/aggregated points

```
The first two modes draw the attractor just draw the whole attractor starting at time 0 upto the defined
time. The first mode draws only the single points, which makes it look more clearly, the second mode
draws a line between the last point and the new one.
The third mode is a little more specific:
In the theoretical chapter there was mentioned, that little differences at the beginning lead to
totally different results (butterfly-effect). This can be visualized with this mode.
At the beginning a 'cloud' of many points, which have almost the same position, is shown.
Then every single point of this cloud is erased, its new position calculated, and drawed again.
After a little while you see, that the cloud, which appeared as a single point, divides itself
and the points slowly go their own ways, distributing all over the attractor. This visualizes, that
a mathematical forecast of the position of a single point after a while isn't possible, because
in practice the exact position at the beginning can't be determined (there are always rounding-errors).
Every little inaccuracy at the beginning has after a while an unforeseeable effect. The only thing, one can say, is,
that the point is somewhere on the attractor.
- With the gadget 'distance' you can define the average distance from one point to another at the beginning, so it defines
the radius of the 'cloud'. The closer the points, the longer it takes, until the butterfly-effect takes
place.
- With the gadget 'Points' the number of points in the cloud is defined. This value is determined
mainly by the power of your computer and the gfxboard you have installed, because
many WritePixel take place...
- This mode makes most fun, if the cloud consists of many points. But this needs much power, perhaps something like
a 200-Mhz-68060. To avoid this, I've added a gadget, which enables a mode, in that the program
draws directly into the bitplanes. This is much faster than a WritePixel, but gfxboards can't handle it.
This mode overdraws all, whatever is put over the window, as example another window or the activated menu.
So pay attention. If you enable this mode, then make sure, that the window is totally visible.

Theory:

                Chapter 2.3.4.1
```

1.33 2.3 Fractals --- 2.3.5 Plasma

```
2.3.5 Plasma

2.3.5.1 Theory
```

Plasma is nothing other than a 2-dimensional Brownian motion. A 1-dimensional Brownian
motion can be made like following: Guess you have a point (as the base), then you
take a random number. Take care, the random number generator must have N(0;1) normal
distribution. Then you horizontally go a step from your point to the right and according
to the random number up or down, dependent on the sign of the number. Now you obtain
a new point, and you are able to repeat the last steps, i.e. a new random number, make
a step to right and up/down, etc... The result is something like a cut through a
mountain, a zigzag-motion once up and once down.
There are several other algorithms for creating a Brownian motion. One other I want to mention, because the
2-dimensional variant of it I use in the program:
Guess you have two points and you want to create a Brownian motion between the two.
Then you take the two points, draw a (virtual) line between them and mark the midpoint
of the line between the two points. Then you take a random number, multiply it with
a value, dependent on the dimension you want and dependent on the length of the interval
of the two points. Then you displace the midpoint according to the number you got.
You obtain 3 points and two (virtual) lines, and with them you act accordingly. This
algorithm can easily be expanded to create a 2-dimensional Brownian motion. Guess you
have 4 points, which form a rectangle. You take the midpoint, take a random number and
displace it according to the random number. Then you displace the 4 other points
in the middle of each line between 2 corners. You obtain again 4 rectangles and
you can act accordingly...
It’s this last algorithm, which is use in my program.

1.34 2.3 Fractals --- 2.3.5 Plasma

2.3.5.2 Parameterwindow 1

Sigma
- A random number is needed at every midpoint-displacement. This number is multiplied
with another number, which depends on the dimension and on the length of the interval.
This number is the base of this multiplicator, i.e. at the first midpoint- displacement
the random number is multiplied with this number, the successing midpoints of the
smaller rectangles are multiplied with parts of this number. The exact ’parts’ depend
on the wished dimension.

H

- This number determines directly the dimension of the object. The resulting dimension
is 3-H, i.e. if H=0.9, then the dimension is 3-0.9=2.1, so it's a rough area. If
H=0.1, then the dimension is 2.9, so it's a very rough area, which is locally almost
a space. This object you could imagine as a mountain with very much and very steep
zigzags, almost space-filling...

ColorMult
- The resulting value is multiplied with this number, the result is interpreted
as the colorindex. This parameter has a similar effect like 'Sigma', but it doesn't
affect the values in the buffer, it only effects another interpretation of the values
in it. So the Plasmafractal doesn't need to be calculated again, it only needs to
be interpreted once again according to the new values. So this saves lots of time,
the fractal just needs to be drawn again...

Seed
- Because this type works with random numbers and a seed of random numbers is deterministic, if
calculated by the computer, it's necessary, to define a startvalue of the seed. The same value results
in the same random-number-seed.

Theory:

Chapter 2.3.5.1

1.35 2.3 Fractals --- 2.3.6 Lyapunov-Space

2.3.6 Lyapunov-Space

2.3.6.1 Theory

The Lyapunov-Space is similar to the bifurcation diagrams. There a formula was used, which
describes the development of the population. In dependence of the growth rate there was shown,
whether a balance exists in the population and if there is one, what type of balance (the lenght of
the eventually existing cyclus).
The Lyapunov-Space now has 2 growth rates, which alternate with each other in a by the user defined
manner.
Look at the sequence AAABB as example, let A be 3 and B be 2 (the two growth rates).
Now this means, that the population of the hares grows in one year with the growth rate 3,
in the next year with 3 too, and then in the next year also with the growth rate 3, (the sequence has
three leading A's), then suddenly with the growth rate 2, then again with 2, and then the sequence starts again
with a growth rate of 3 etc.

```
Now you examine this model for all different values of A and B. A is drawn ↩
   horizontally and B vertically.
Now we must decide, what we should draw at a concret position. Of course, we draw ↩
   a pixel at this place, but of what color?
With the bifurcation diagrams there were mainly two classes of points:
1. class: Values of the growth rate, which lead to a cyclus of any concret finite
lenght.
2. class: Values of the growth rate, which don't lead to a balance.

Here we take this classification and color a pixel accordingly. Now the question
remains, how we can decide, whether a concret point (with concret values for A and ↩
    B)
leads to a cyclus or diverges.
Well, let us examine the formula f(x)=a*x*(1-x)
We define, that there is a balance, if the average of the absolut value of the ↩
   derivation of f(x) is less than 1, otherwise there is chaos.
But we must give a chance to the point, to be attracted by a cyclus, like we had ↩
   to do previously with the bufurcation diagrams.
In practice we make the logarithm, so we have the following algorithm:

X=0.5 ; the population start value...
    ; Now we must give a chance to X, to go to an attractor...

FOR N=1 TO 4000
    ; R is A or B, dependent on the sequence...
   X=R*X*(1-X)
NEXT N

    ; upto here there should all be clear

Sum=0
FOR N=1 TO 6000
    ; R is again A or B, dependent on the sequence...
   X=R*X*(1-X)
(add the absolut value of f'(x)=R-2*R*X, but take the logarithm of that)
   Sum=Sum+Ln|R-2*R*X|
NEXT N
Sum=Sum/6000    ; build the average

Well, the result, contained in the variable 'Sum', is the average of the Ln|R-2*R* ↩
   X|, the logarithm of
the derivation of the formula f(x) and represents the rate, with which the ↩
   population grows. It is called Lyapunov-exponent.
If the average is negativ (that means, that |R-2*R*x| is average less than 1), ↩
   then
balance takes place, otherwise chaos.
Chaos we color with a single color, mostly black. I've tested to color the chaos,
but I've found out, that it's really intelligent, to call is chaos...
If a value of less than 0 is the result (balance, in practice we get values mainly ↩
    downto -5),
then we multiplicate the number appropriate, cast it to an integer and color the ↩
   pixel
with this number. That's all...
```

1.36 2.3 Fractals --- 2.3.6 Lyapunov-Space

2.3.6.2 Parameterwindow 1

Formula
- These formulas are identical to the formulas at the bifurcation diagrams, except that formula number 3 is missing, because
there I didn't know, what to do with the derivation.

ExpMin
- Here you can define the minimal exponent. The colors are automatically adequat distributed
to the defined Lyapunov-exponent range. All values, which result in a smaller exponent,
are coloured with color 4.

Start x
- Everybody, who looks at the Lyapunov-space, regocnizes these spikes, which cross each other.
It's very strange, that the position of the spikes, I mean, whether spike number 1 is behind or in front of spike
number 2, depends on the initial value of x (in the algorithm we had initialized x with 0.5).
This initialization you can set with this gadget.

Sequence
- Here you can set the sequence of the two growth rates. In order to actually take place,
you have to press the return-key (or help, or tab...).

Passes
- In order to visualize more quickly the Lyapunov-space, one can artifically lower the resolution, like it's made
at the julia- and mandelbrotsets. Everybody, who now is terrified, because he thinks, that I take this method
in order to make the calculation faster (like I do with julia/mandel), I can calm:
All I do, is, as a preview lowering the resolution. By choosing 3 passes the Lyapunov-space
is actually drawn more slowly as with 1 pass (but you won't recognize it too much), but
you get more quickly an impression of what it looks like and can zoom in further or change a
parameter.

Chaoscolor
- Name says all, or not...?

Stabilization
- This value defines, how often the formula is first iterated, until the exponent is calculated. This gives a chance to the point,
to be attracted by an eventually existing cyclus.

Iteration
- Here you can define, how often the formula after the stabilization has to be iterated, in order to calculate the

Lyapunov-exponent. I recommend to first setting this value to a low number (perhaps 20, because then the space is drawn faster), then to increase it. After that, you'll
see, whether the picture changes a lot...

Area
- This should be clear...
A is drawn horizontally, B vertically

Theory:

Chapter 2.3.6.1

1.37 2.3 Fractals --- 2.3.6 Lyapunov-Space

2.3.6.3 Datawindow

In the datawindow the to the mouseposition corresponding growth rates A and B are displayed, and the Lyapunov-exponent is
calculated again.

Theory:

Chapter 2.3.6.1

1.38 2.3 Fractals --- 2.3.7 IFS

2.3.7.1 IFS

Nowadays almost everbody is talking about fractal image compression. This 'new' method promises incredible
packing rates. If one believes the others out in the world, then it should be possible to describe
whole images just with a small set of numbers. The basic idea for this pack algorithm is an iterated function
system, short IFS. To be exact, Julia- and Mandelbrotsets are also iterated function systems, but this chapter doesn't
care about these special fractals. Here I will only have a short glance at linear iterated
function systems.
As it was the case with the Julia- and Mandelbrotsets, here again one is interested
in the 'attractor'. You have seen already (at least I hope so), that this attractor
can be very complicated, just remember a few pictures of the Mandelbrotset. The special thing about the
(linear) IFS is, that one is not only able to calculate the attractor from a given IFS. One is also
able to reconstruct an IFS from an attractor. So you have a picture, assume that it is an attractor of an unknown IFS, reconstruct

```
the IFS, save the IFS (which normally needs only a small set of numbers) and if ↩
    you want to look again at the picture, you just calculate
the attractor of the saved IFS and get a picture, which is similar to your ↩
    original picture.
This is the basic idea behind fractal image compression. One only has to choose ↩
    the right functions, so that the attractor is similar to
the picture on wants to compress. Sounds quite easy, doesn't it? Well, now, how ↩
    exactly
does it all work?
An IFS consists of several functions, every single function is an affine ↩
    transformation, that means,
a function, which just rotates, moves around or streches a given object. Every ↩
    such affine transformation
has a value assigned, which is interpreted as the execution probability, that ↩
    means, this value determines, how
often this function is applied to a point compared to the other functions in the
system. Such affine transformations can be represented simply by a matrix ( ↩
    responsible
for rotating and stretching) and a vector (responsible for moving the object ↩
    around)
.
Now, how is a picture, the attractor of an IFS, calculated?
1. One starts with a point, which surely lies onto the attractor. For example, one ↩
     can take the fixed point of
the first transformation.
2. Choose a function of the IFS, such that the given probabilities are fulfilled
.
3. Apply the chosen transformation to the last point. The result is a new point of ↩
     the
attractor. Go to step 2...

Well, after some time the attractor will appear.

Now lets come to the question, why and especially how it is possible, to construct ↩
     an IFS to a given picture.
There exists a mathematical proposition, called the 'Collage Theorem', which ↩
    essentially says, that this
construction is possible.

Given an IFS with the transformations w1 to wn, additionally a set T. Let s be the
greatest Lipschitz-constant of all the transformations w1 to wn, which must be ↩
    smaller
than 1. Well, s should be, in my own words, the maximal factor of all the ↩
    contractions.
Every transformation is some sort of rotation, stretching and movement. Rotation ↩
    and movement doesn't
affect the distances of the object, just the stretching affects the distances of ↩
    the object.
So s should be the maximal stretching factor of all transformations. Now in this ↩
    case this is the
Lipschitz-constant. And it has to be smaller than 1.
Now let me continue with the theorem:
The Hausdorff-distance between T and the union of w1(T) to wn(T) should be smaller ↩
     than
a given epsilon. Then the Hausdorff-distance between the attractor of the IFS and
T is smaller than epsilon/(1-s).
```

Well, the Hausdorff-distance of 2 sets is, to be quite inexact, and, to be precise , wrong,
but easier to understand, the 'normal' distance between 2 sets.

Well, now let me explain, what this theorem essentially says:

Given a set T, this could be perhaps a picture (should be a black-white-picture ...)
Then the transformations of the IFS, w1 to wn, are chosen, so that the union
of the sets w1(T) to wn(T) covers quite exactly the whole set T, that means, the union of the sets, which arise, if you separately apply
w1, w2, ..., wn to the set T, covers the picture. Remember, the transformations are functions, which just rotate, stretch and move an object around.
So all this last sentence says, is, that the functions w1 to wn should be chosen, such that the whole picture T
is built of small copies of itself, just like a collage...w1(T) is a rotated, stretched (with
a factor smaller than 1!) and displaced copy of the original set/picture T, so is w2(T),
w3(T),...wn(T).
Well, then the condition of the theorem is approximately fulfilled, i.e. the Hausdorff-distance
between T and the union of w1(T) to wn(T) is smaller than epsilon. This epsilon of course is determined
by the quality of the collage. Then the theorem says:
The attractor of the IFS, which one can calculate out of the transformations, is
similar to the original set/picture T. To be more exact, the Hausdorff-distance between this attractor
and T is smaller than epsilon/(1-s). Now you see, why the Lipschitz-constant must be smaller than 1.
And you can see even more: If you have a very small Lipschitz-constant, then (1-s) is near 1, but then of course you need many transformations to
cover your picture, because the Lipschitz-constant is the stretch-factor, so you have to build the picture T out of many very small copies of it.
So if you want to make a very good collage, then the compression ratio isn't that good.
Also notice, that the calculation time goes up, if 's' is small. It makes sense, to choose 's'=0.25, so to build the
whole picture of quarters of itself.
As you can see, too, the number of differences between the attractor and the original picture depend on epsilon. So if you make a bad
collage, then this epsilon is big...

Well, this is the idea for fractal image compression. One simply has to find out the
transformations w1 to wn.
One could implement the following algorithm:
Given a picture, this picture is then cut out and used as a brush, just as
in any paint program. The user then has to rotate, move around and stretch
this brush, so it covers a part of the original picture. Then the user has
to fix it. the first transformation is completed (one simply has to write down
all done actions like moving, rotating and stretching). Then one fetches
another copy of the original picture, again moves, rotates and stretches it,
so it covers another part of the original picture. It then should be fixed, too,
and so the second transformation is ready. The user continues like this, until the
original picture has been covered by some smaller copies of itself, just like a collage.

```
Let me note one thing: The parts of the copies may of course overlap, but this ↩
    just increases
the calculation time, so it should be avoided.

Of course I already tried to write such a program. Unfortunately the resulting ↩
    transformations
weren't good. I don't know why it didn't work, it just has to work. Well, perhaps ↩
    in some time
I manage it to write such a program.
```

1.39 2.3 Fractals --- 2.3.7 IFS

```
                2.3.7.2 IFS-Parameter 1

Most of the parameters should be self-explanatory.
Formula
This listview lets you choose the formula, which one has created using the
                Formula editor
                .

Parameter
Right beneath the listview one can choose upto 4 parameters, which eventually are ↩
    used by the
formula.

Iteration
This number determines the number of points of the attractor, which should be ↩
    calculated
.

Colormode
This gadget lets you specify the coloring mode.
There are 3 possibilities:
1. Transformation: The color of the point is determined by the last applied ↩
    transformation
.
2. Probability: Just like in 1: In order to get the transformation, a random ↩
    number has to be calculated. Here this random number defines the number of the ↩
    point. This of course
leads to similar results like in 1.
3. Measure: This is the best mode. The greater the measure at a point, the greater ↩
     the color
register.

Area
Should be clear...

Angle
An IFS is, at least in ChaosPro, always a 3 dimensional IFS. So these gadgets ↩
    exist, which let
you specify the view angle.
```

1.40 2.3 Fraktale --- 2.3.8 L-System

2.3.8.1 L-System

Well, the correct name for this fractal should be DOL-system, or, to say it in english,
deterministic contextfree Lindenmayer systems. Deterministic, because no random plays any
rule in opposite to perhaps IFS, contextfree, because all this stuff is about contextfree
grammar (that's the jargon of the informaticans, it only means, that there are rules for
replacing text, which have to be applied regardless the context, i.e. rules of the form
'character <c> has to be replaced by the string <string>' and not of the form (contextsensitive)
'<string1> has to be replaced by the string <string2>'). Well, I think, this was enough stuff about
the name.
What's all this stuff about?
An L-system consists of an axiom, simply a string and a set of rules, all of which have the
form 'character <c> has to be replaced by the string <string>'. Now one starts with the axiom as the initial string.
Then the first character is examined. If a rule exists, which says, that this character should be replaced
by a string, then it should be replaced by this string. If no rule for this character exists, then just leave the character
as it is. Then have a look at the second character, replace it by a string, if a rule exists, or simply do nothing, if no
rule exists. Continue this way, until the end of the string is reached. Then the first iteration is finished. The result
is most likely a string, which is quite a bit longer that the string before the iteration.
Now the second iteration starts with the new string, one starts again at the first character, replaces, goes
to the second, the third, etc. until the end is reached, then goes on to the next iteration, starts again at the
first character, etc., the 4th iteration, the 5th, etc. until MaxIt is reached. As you can imagine,
values of less than 10 for MaxIt are very popular, because the string will grow exponentially...
At the end one has a huge string, which doesn't look like a fractal. But now the magic starts:
The whole string is interpreted as a series of drawing commands. A drawing command
consists normally of a single character, like 'F', which simply draws a line. The resulting image is a fractal, at least, if
one would iterate infinitely often.
The Lindenmayer-systems are of importance in conjunction with computer generated natural objects, because the results
show some similarities to objects, which appear in nature, mostly plants. Unfortunately these deterministic contextfree
systems aren't that powerful. It's very difficult to create realistic plants with them.
If one wants to create such plants, one has to specify a huge amount of rules to describe simple

plants. Due to this, there exist some enhancements to DOL-systems, which then are contextsensitive and
additionally not deterministic.

But not lets give an example of an L-system, just to make clear, you have understood the basic
idea:
Well, lets define the Koch-curve: Everybody should know this fractal, additionally it's ideal
for creating via L-systems:

Keep in mind some drawing commands:
F, which draws a line of given lenght into the current direction
+, which changes the current direction, say about 60 degree counter clockwise..
.
-, which changes the current direction, say about 60 degree clockwise...

Lets start with the axiom: F
Now lets define one single rule: F has to be replaced by F-F++F-F

This is exactly the definition of the Koch-curve: Every line (F) has to be replaced by an
object, which is the line, but the middle third part of the line is replaced by a triangle. Just
draw this F-F++F-F, and you see this replacement. It is obvious, that this very simple
L-system creates the Koch-curve.

Generally the whole kind of interpreting the drawing commands is very similar to the
turtle-graphics. But who knows nowadays, what turtle graphics is? In former times this
kind of drawing was very popular. One has to imagine a virtual turtle on the screen,
which can do some actions. It can move forward, draw forward, it can rotate about an
angle, it can change the drawing color, etc.

For an explanation of all the drawing commands see the chapter about the L-system formula
editor.

1.41 2.3 Fractals --- 2.3.8 L-System

2.3.8.2 LSystem-Parameter 1

The listview lets you choose the L-system formula, formulas can be changed in the
formula editor window for L-systems.

Right beneath the listview there are 4 parameters, which are currently unused. Below these gadgets
there is a gadget, which lets you choose the number of iterations to perform.
Normally you should use values between 3 and 15. I recommend to start with about 3, then to increase it step by step.

Below the listview you can choose the area. Other programs calculate L-systems in another way. They first completely generate the
string, then parse it to get the size of the fractal, and then choose the area accordingly. Of course this has be
disadvantage, that the user doesn't see anything for quite some time, so I don't like
it.

1.42 2.3 Fractals --- 2.3.9 Diffusion

2.3.9.1 Diffusion - Theory

Well, what should I tell you now?
Some effects in nature remind people heavily on fractals, for example lightning. Or diffusion effects,
for example, if one mixed oil and water. These two liquids have a very intricate bound between each other, which also remind
to lightning. Well, such a diffusion one can simulate by the computer. This is done in the following
way:
One starts with an object on the screen, for example a point. Then a new point suddenly appears anywhere on the screen.
This new point one moves randomly around, such that it performs a random walk, a brownian
motion. To be more exact, one chooses randomly 0, 1, 2 or 3, and then moves the point up, if he choosed 0, right, if he choosed 1,
etc.

Of course, this can be continued for some eternaties. So one needs an abort criteria. Finally there should
be a diffusion fractal. Well, as soon as the point arrives in the neighbourhood of a fixed point, it adheres to it and
so gets fixed. Then a new point is created, same procedure is performed, etc. The resulting image
then is some sort of diffusion.

1.43 2.3 Fractals --- 2.3.9 Diffusion

2.3.9.2 Diffusion Parameter 1

Well, what parameters can be chosen?

Type
Here you have 2 possibilities:
1. Diffusion: Calculates a diffusion fractal
2. Random Walk: Just shows a 'Random Walk', i.e. how a point moves around randomly
.

Shape
You can choose the starting shape of the fixed points. Currently 4 possibilities are
available. If the starting shape is a line, then it should be clear, that the moving points adhere

at this line. So the resulting image should like moss, i.e. many lightnings out of a
line.
a) Point: Just a point in the middle of the window
b) Line: Just draws a line
c) Rectangle: Draws a rectangle
d) Random: Randomly draws some points

Show Walk
If checked, then the walk of the point is shown, otherwise just the result is shown,
i.e. the point, as it gets fixed...

Seed
Determines the initial value for the random number generator.

Colormode and Coloradd
a) Neighbour: The color of a point is the color of the neighbour point + Coloradd
b) Time: The color of a point is determined by the amount of time, the point moved around.

Sticking probability
Values between 0 and 1 are legal. This value specifies the probability, with which a point, which encounters a fixed point, also stays fixed.
I don't think, you understood this, did you? Well, upto now you assumed, that a point, which moves around and suddenly encounters a fixed point in the neighbourhood, adheres to this point and stays fixed, too. Well, but this is not a must-be. A point needn't stay fixed, it can move around until it finds another fixed point. Perhaps then it stays fixed. And this is, what this 'Sticking Probability' means: How probable is it, that a point, which encounters a fixed point, adheres to it? Now all should be clear ...

Boxsize
It takes much too much time to calculate a real diffusion fractal. So I had to speed it up.
The program calculates the smallest square around the set of all fixed points. It then calculates a bigger square, which is just 'Boxsize' pixels bigger (i.e. the side lenght is longer) than this just calculated square. And now this greater square is the region in the window, where all happens: All new points appear in this square, and not anywhere in the window. You can imagine, that this speeds things up quite a bit. And yet another 'hack' was done: As soon as a point leaves this square, it is thrown away, a new point inside this square is generated. It would take too much time, until the point, which has left the square, comes back into the square, so a new point is calculated. It shouldn't matter that much, because it's totally random, where this point encounters the square again, so it should be be same to just calculate a new point and place it into the square at random.

1.44 2.3 Fractals --- 2.3.10 Brown

2.3.10.1 Brown - Theory

Now lets come to the next fractal type, the Brownian Motion. Don’t panic, I don’t bother you with
a view from the probability theory, so I don’t tell you something about infinite paths, non-differentiable
at any point and so on.

Basically I don’t have to explain anything to you, because ’Plasma’ is a two dimensional Brownian Motion.
Such a Brownian Motion has some very pleasant properties. For example you can draw a 1 dimensional Brownian Motion onto
the screen and it just looks more like a music sample than a mathematical object. You can change the dimension of
the fractal by adjusting a parameter. Of course you wonder, how it sounds, if one decides to play this
brownian motion. ChaosPro can do this, don’t panic.
After some examinations people have found out, that music shows some similarities to brownian motion, that means, you
can ’play’ a brownian motion and it makes sense...
You shouldn’t expect to hear nice sounds or nice music. You only hear something, that’s all.
The music sounds a bit strange, somehow from an alien planet, from a foreign culture.
And of course you can expect, that typical elements of music, such as refrain, themes, etc.
appear with brownian motion. So please keep your CD’s from Mozart, Beethoven, the Beatles,
etc...

1.45 2.3 Fractals --- 2.3.10 Brown

2.3.10.2 Brown-Parameter 1

Scaling
Specifies a factor, which is needed for the calculation. It must be between 0 and 1. The closer to 1, the longer
it takes for the fractal to finish. This parameter doesn’t have anything to do with the graphical
representation, it determines dierectly the number of passes it needs to calculate the fractal. This parameter is here due to the
algorithm I use for calculating the fractal.

H
Determines the dimension, Dim=2-H...

Samples
Specifies the number of points, which are calculated for the Brownian Motion
.

Play & Stop
Starts/stops playing the Brownian Motion. The height is defined by ’Rate’, the type (music or sample)
by ’Mode’.

Mode
This gadget determines, how the Brownian Motion should be played. There are 2 possibilities:
Sample or Music. The first one just creates a sample, which then is played, and the second one plays
music, i.e. it plays tunes.

Rate
This gadget lets you specify the height of the sound.

1.46 2.3 Fractals --- 2.3.11 3D-Ansichten

2.3.11 3D-Views

2.3.11.1 3D-Introduction

The 3D-modul is a kind of modul, which could easily be converted to an external modul.
It takes no care, to what fractal (if anyhow) the data belong, which it gets from
an array. This method of course has a few disadvantages, because so this modul can't
get additional values, if it would need them in order to increase the quality of the view.
But an advantage is, that in a later version of this program another person but my can
build an array of heights and then give it to this modul, which forms something threedimensional out of it.
So you save a lot of time. You needn't write routines of your own. Additionally, I can build in a
routine (or you write your own), to use such a heights-file from SceneryAnimator or
a similar program. Well, much is possible, but what I'll implement, you decide with your
reactions.

1.47 2.3 Fractals --- 2.3.11 3D-Views

2.3.11.2 3D-Parameterwindow 1

Projectionmode
There are 2 modes available:
1. Orthogonal: This is the favorite method. Here every point is simply projected onto
a 2D-plane by an orthogonal projection. The distance doesn't affect the picture, it's
meaningless. The implementation of a horizontal angle isn't possible, I tried it.
Instead use the rotation of the 2D-fractal. This mode always draws the picture in
the best possible quality.
2. Perspective: This is the old, bad method, only implemented again due to the wish
of my favorite betatester. Here the 3D-object is projected onto a 2D-plane, just
like the human eye does it. Here more things can be adjusted. The big disadvantage

of this algorithm is, that the whole algorithm is very complicated and can't be simplified for the computer. This makes this mode very slow. The whole code also is very long and complicated, so enhancements are really a pain.

Drawingmode
Only with Projectionmode=Projection...
Here you can choose, how all the points are to be displayed:
1. Points: just draws the points
2. Gridlines: draws lines between the points
3. Rectangles: draws a rectangle out of 4 points
4. Spikes: draws spikes starting from the ground.
5. Mosaic: simply draws small rectangles of size 2x2 at every point
6. Cross: draws small crosses at all points

The best thing is to try it...

Distance
Only with Projectionmode=Projection...
Name says all...

H-Angle
Only with Projectionmode=Projection...
- The horizontal angle, from which the observer looks at the object. It corresponds to the degree of latitude on a globe.

V-Angle
- The Observer always stands right in front of the fractal and looks from a certain height onto the fractal. The height is defined by the vertical angle, which corresponds to the degree of latitude of the earth.

Light
- If checked, a lightsource exists , which is infinite far away. The position of the lightsource is defined by the horizontal and the vertikal angle, which correspond to the degree of longitute and latitude of the earth. Suppose, Amerika lies at the H-Angle 0, then Europe lies at about 90 and Japan perhaps at -90. If light isn't checked, then the original colors are used.

Intensity
Diffuse
Ambient
Reflection
- The brightness of an area is defined by these values:
* 'Intensity' determines the intensity of the light source, so that you now slowly can switch on a light source (perhaps you calculate an animation). This value must be between 0 and 1.
* 'Ambient' is a number between 0 and 1 and determines, how much light falls on every area, independent to whether light from the lightsource falls on it. A value of 1 doesn't make much sense, because then every area would be drawn with an intensity of 1 (brightest light).

```
*'Diffuse' determines, in what proportion the light from the lightsource stands
to the reflected light. A value of 0.8 means, that 80% of the intensity of an area ↩
    is defined by the angle,
in which the light from the lightsource falls on it, and 20% of the intensity of  ↩
   the area by the angle, which
is enclosed by the vector of the reflected light and the vector from the observer.
To be more general, it defines, whether the 3D-picture shines due to reflected  ↩
   light or due to light from the
lightsource.
* 'Reflection' determines, how strong the areas reflect the light, which falls on  ↩
   them. 1 means strong, 2 means less strong,
0.5 means very strong, etc.

GridX and GridY
Only with Projectionmode=Projection...
- Here you can define the resolution of the X- and Y-direction. Smaller values  ↩
   result in
a speed up of the drawing, but of course they lower the quality...
```

1.48 2.3 Fractals --- 2.3.11 3D-Views

```
2.3.11.3 3D-Parameterwindow 2

DeltaX/Y
- The object itself is drawn around the nullpoint. In order to move it, these 2  ↩
   sliders
have to be used.

DeltaZ
- With this gadget you can move the object up or down.

Invers
- Inverts all heights. This makes out of the Mandelbrot-mountain a Mandelbrot-
valley...(I prefer valleys...)

Autoadjust
- Tries to move and size the 3D-fractal in a way, so it fits entirely into the
window.

FrontMult
BackMult
- Determines, with what numbers the heights in the front or in the back have to
be multiplied. Normally you set these 2 numbers to the same value. But if you want
to give the object more plasticity, I recommend raising the BackMult-number a bit.
The heights in the mid are multiplied with (FrontMult+BackMult)/2, so it's going
from FrontMult to BackMult in a linear manner.

Slope
- This value defines, how steep the mountains and valleys should be.
Smaller values ==> less steep
Greater values ==> more steep
The function is:
x^(1/Slope)
,where x stands for the height to be transformed. That means, if Slope is equal
to 2, then the function is x^0.5, which is the square root of x.
```

YStretch
This values defines the factor for the y-direction of the object. This direction
points to the 'back' of the object. If the 3D-view appears to be a bit too 'short ↩
    ',
then you should raise this value to greater than 1.

Water
Plateau
- 'Water' defines the waterlevel. All heights, which would be lower than this ↩
    value,
are considered to belong to an 'ocean' and are set to this height. The color of ↩
    the ocean is
determined by the angle the light falls onto it.
- 'Plateau' defines the plateaulevel. All height, which would be greater than this ↩
    , are considered to form a
plateau with exact this height. Again, the color of the plateau is determined by ↩
    the angle the light
falls onto it.

1.49 2.3 Fractals --- 2.3.11 3D-Views

2.3.11.4 3D-Parameterwindow 3

colors to use
Here you define the colors, which are to be used for coloring the 3D-fractal.
The first 3D-color is considered to be 'black', i.e. it is used for an area,
onto which the light doesn't fall, and the last 3D-color then is 'white', i.e. ↩
    used for an area, onto which
the light directly falls.

Background
At the beginning of the 3D-view-drawing the whole window is cleared with this ↩
    color.
If areas remain free, then they appear in this color.

Dithering
You can choose one of 3 modes: The first, no dithering, the second, which tries
to double the number of colors trough dithering, and the third, which tries to
get 4*NumberOfColors through the aim of dithering.

ExtBuffer
If you make a 3D-transformation of a 2D-fractal, then of course only the buffer ↩
    values
can be transformed. But this most likely results in an image, which seems to be ↩
    cut in
the front and in the back. 'ExtBuffer' now allows you to increase the (vertical) ↩
    size
of the buffer in percent, so more buffervalues are available. For a 2D-fractal ↩
    this
is totally useless, i.e. set ExtBuffer to 0. If you make a 3D-transformation, ↩
    values at about 30 to 50 may be useful
(30% to 50% of the original 2D-buffer-size added).

Saturation

```
Value
These values only have an effect, if you save the 3D-picture in 24 Bit. In the 3D- ↩
    buffer the
original colors and the lightintensity is stored for each pixel. These 2 gadgets  ↩
    determine, how
the information of the color and the light are combined to form the resulting  ↩
    color.
Saturation: Determines, how much influence the lightintensity has on the  ↩
    saturation of the original color.
The range is from 0 (light doesn't affect the saturation) until 100 (saturation  ↩
    comes totally
from the light-intensity).
Value: See at 'Saturation' 3 lines above, but now the value of the color will be  ↩
    affected by the light.

Normally one sets the value to 100 and the saturation to 0, this means the  ↩
    original color is taken,
converted into the HSV-colormodel, the value replaced by the light at this place,  ↩
    then converted into RGB-Format
and stored.

Riemann
(not implemented!)
- Fractals are almost always representations of what happens to complex numbers,  ↩
    if they are iterated
in a specific manner. Every screenpixel corresponds to a complex number. Now  ↩
    Riemann
has thought out another representation for the complex numberplane. Normally one  ↩
    speaks from
the comlex numberplane, the one axis is the real axis, the other the imaginary  ↩
    axis.
Riemann now has transformed this plane onto a spheare. This spheare has the radius ↩
     1 and
touches with its southpole the complex numberplane in the origin, the nullpoint.
Now, how is the new point calculated from the old one? If you have a point of the  ↩
    complex numberplane, you
simple have to draw a line between this point and the northpole of the spheare.  ↩
    This line
will run through the surface of the spheare. And this point is the new one, which  ↩
    corresponds to the old.
Well, all's fine, every complex number is matched to a single point of the surface ↩
     of the spheare, but
there's one exception. The infinity. It consists of infinte many points. But all  ↩
    of these are
matched to a single point on the spheare: The northpole.
Perhaps this is it, why in the theory of the fractals almost always the infinity  ↩
    is
considered to be an attractor, because it's on the Riemannspheare just a single  ↩
    point.
Well, this program can draw the 3D-representation of the Riemannspheare. But it's  ↩
    not
the standard Riemannspheare, because it touches the complex plane in the origin  ↩
    and has a fixed
radius of 1. This program calculates itself the point, at which the spheare  ↩
    touches the plane, because
otherwise it wouldn't look very good. The radius you can set. If it would always
be 1, then it would look not like it is supposed to look.
```

Just imagine, what it would look like, if you would zoom into the fractal and then see a
representation of this. For people with less fantasy: There are much too less values, so
if you have the whole spheare in front of your eyes, then eventually there is just the little area
visible, which corrresponds to the fractal. Now if one thinks, that somebody simple have to
zoom the spheare, then the area of the spheare would look like a plane, just like
a landscape don't look like something on a spheare, although it's on a spheare,
the earth.
You can let the program set the radius for you. Simply click onto the corresponding
gadget, and the program calculates a radius, so the whole fractal will happily fill
the spheare...

1.50 2.3 Fractals --- 2.3.12 Wizzardwindow

2.3.12 Wizzardwindow

Dependent on the fractal type, this window contains different data.

Information
Here you find the duration of the calculation process, eventually how often you have zoomed in and
the number of bytes, which are used by the buffer.

Analyze
Here you find the profile of the iteration values, i.e. how may points of what iteration
depth exist. This is very useful, because you see, whether the iteration depth is too large.
Also good, in order to get optimal values for the colormapping, because you can decide more easily,
how the colormapping function has to be. The higher the bar in the iteration profile graph,
the higher should be the first derivation of the colormapping function at the same
place.

Heights
Well, one of the biggest problems with ChaosPro is to find suitable values for the 3D-transformation.
If you choose this mode, then the program draws a graph, in order to draw the actual transformation function. This way you immediately see, what
effects the parameters have.

1.51 2.3 Fractals --- 2.3.13 Commentwindow

2.3.13 Commentwindow

Well, sometimes it's useful to give some hints to fractals. Also some people are pleased to see their name appearing with fractals, which

they have found.
Ok, you see the use of the window and the gadgets immediately, so I don't want to waste my time by
explaning every gadget...

1.52 2.4 Menus

2.4 The Menus
2.4.1 Systemmenu

Other Menupoints:

Fractalmenu

Fractalwindows

Windows

Extras
Data load/save
- This menuitem loads/saves the fractaldata out of/into a file. If the file is saved to
the directory ChaosPro/FractPic, then it will automatically be loaded and it'll appear
in the listviewgadget, which contains all the fractals. If a file is loaded at runtime,
then it will be added to the list, which is already shown in the listview of the
PicTask-window.
When saving the program takes care, whether the 3D-fractal window is open. If this is the case,
then it saves it as an active 3D-fractal, i.e. when this fractal is calculated the next time, both windows, the
2D- and the 3D-window are automatically opened.
Starting with V2.0 ChaosPro is able to load Mand2000, Mandelmania and compatible fractal
data files.
ChaosPro now saves the data files automatically into every IFF-ILBM picture, which is saved, so
you can reload an IFF-ILBM picture and continue working with the fractal.

Picture/Load
If you have the datatypes.library (OS3.0 or higher), then you can load pictures into ChaosPro.
If you choose this menuitem, then a requester appears, asking you for a picture to load. After that
the datatypes.library will be opened and the picture will be loaded into the fractalwindow of the currently active
fractal. This window will be resized, so that the picture will suit into it. Due to the use of the datatypes.library
I saved much time, as a side effect you can load every type of picture, as long as the datatypes.library is able to understand
it.
There exists another menu item, which belongs to this loading procedure, the item
'Settings/Misc/Picture Remap'. If this item has a checkmark, then during load time the palette of the picture will be

adjusted to the screenpalette. The loading process then of course lasts a bit longer. I personally don't check this item, because I normally only load fractals,
and there needn't to be a 'remap'.

Save Picture/to Clipboard
- This item saves the picture as an IFF-ILBM-picture. For the fractaltypes Juliaset/ Mandelbrot/ Plasma/ Lyapunov-Space it's possible, to save the 2D- picture
in any depth upto 8 planes (256 colors) independent from the hardware and from the actual screenmode (it will be
saved, not displayed...). You are prompted for your favorite depth after choosing the filename.
This enables the owners of older Amigas to save a picture in 256 colors, and then to convert it
to a HAM6-picture by another program.
Additionally, there were some people, who wished to save 24Bit-images. This is also possible.
When prompted for your favorite depth, there are two more possibilities: 24 (Screen) and 24 (256).
The first one saves the picture with the colors used on the screen, but with a depth of 24 bit.
The program actually calculates more colors than it can display and then calculates down to the
available number of colors. If now there are areas on the screen with the same color, then it's possible, that the program had to
assign the same color to various values, which didn't differ too much. If the the 24bit-image is saved, then
there are enough colors available, so the used palette is 'blown up', and so the correct color is used. This creates a smooth
flow from one color to the next.
Please notice, that when using the iteration-coloring with Julia/Mandel optically the 24bit-image is identical to
the image, which you get, when you save it in , say, 256 colors. If you here really want, what I gess you want,
many many colors, smooth flows from one color to the next etc., then you have to use the
CPM-method.

The other possibility, 24(256), takes the whole palette of 256 colors, blows it up , and then saves the image.
This is identical to choosing 256 colors, but now perhaps areas of the same color are replaced by a color-flow to
the next color.

Well, don't be disappointed: If a fractal doesn't contain areas, then the 24bit- image doesn't look
(recognizeably) other than the one with, as example, 6 planes. So to save a plasma -fractal with a high granulation is somehow not
intelligent. Think over it, and you'll recognize, why.

With the Bifurcationdiagrams and the dynamic systems and also with the 3D-views of the fractals all these additional possibilities don't exist. The fractal is
saved exactly like it's displayed. Exception: If the fractal has a 3D-buffer, then it can be saved in
24 bit.

```
Print
- I thought about implementing a routine for printing fractals for quite some time ↩
    . Finally I came to the conclusion, that
it doesn't make much sense to write such a routine, because it would have to be a  ↩
    really good routine.
One wants to print fractals in the best possible quality, so I would have to use a ↩
     color management system.
But AmigaOS doesn't offer a CMS. Due to this I simply have used the server  ↩
    function of 'Studio' from
Wolf Faust. So I have a great print routine, a color management system, so that
the fractals on the paper look exactly like on the screen. Of course, in order to  ↩
    print, one first
must have 'Studio' from Wolf Faust.
Annotation:
If you choose this menu item, ChaosPro first saves the fractal in the temporary  ↩
    drawer. So the user
will be asked about the depth of the fractal. The temporary drawer is 'ChaosPro:', ↩
     but can be changed
using the tooltype 'TempDir=<Dir>'. After saving the image Studio is called  ↩
    asynchronously, which
then prints the picture and deletes it after printing.

Systeminfo
- This item shows a few informations about the processor/ coprocessor/ gfxchips/  ↩
    priority and
the memory

About ChaosPro
- It shows information about the Author and the version of the program.

Quit
- Explanation necessary?
```

1.53 2.4 Menus

```
                2.4.2 Fractalmenu

Other Menupoints:

                Systemmenu

                Fractalwindows

                Windows

                Extras
                Juliaset
Mandelbrot
Bifurcation
Dynamic System
Plasma
Lyapunov-Space
Diffusion
IFS
```

L-System
Brownian Motion
- These items add a new fractal of the corresponding type to the listview-gadget. They are initialized with
the default values for the type.

Defaultvalues
- If you have changed the parameter of a fractal and now don't know, how to come back to
some good values, then you can choose this item. It sets the parameters to the defaukt values
of the type.

Edit Windowsize
- 'Edit Windowsize' shows the actual size of the 2D-window and lets you input new values. The
maximum size isn't the screensize, but the screensize minus the bordersize. This function
currently can't convert a normal window into a backdrop-window and back. I currently consider
it as a feature rather than a bug, that this window doesn't automatically convert backdrops
to normal windows, because now it's possible to make a backdrop-window and then to size this
window with the new menuitem (of course, it may be somehow confusing, if a window without a border
exists somewhere on the screen..)

Zoom
- 'Scale in' - 'Scale out'
- 'Scale in' is just the same as a doubleclick into the mid of the fractal window.
'Scale out' makes the opposite including the scaling etc.

- 'Box in' - 'Box out'
If you choose one of these items, then you can click anywhere into the 2D- fractalwindow (you may leave the button again...).
Then you carry a frame around. Click again to leave the frame. If you have chosen 'Box zoom in',
then the box defines a new area, which is made bigger, so it fits exactly into the window. If you have chosen
'Box zoom out', then the whole window is projected into the defined box, and the fractal with the new area-values
calculated again.

Undo/Redo
- Unlimited Undo/Redo for every fractal. There is always a buffer of 10KB size allocated, in which
the old values are stored.

Move...
- This moves the fractal, you can achieve the same, if you press the cursor-keys
in the 2D-fractalwindow.

Proportion

```
- if the fractal is heavily distorted, then the proportion of the area-values ↩
    doesn’t fit to
the actual proportion of the width to the height. This item restores the ↩
    proportion by adjusting the
area-values.

Calculation
- Stop/Continue: Because the program runs in a multitasking environment, it is ↩
    possible, that
you calculate more than one fractal at the same time, but want to finish one ↩
    specific
fractal as quick as possible. This item stops the calculation of the active ↩
    fractal. The task
is put to sleep (by a totally systemconform method...) and can be waked up by ↩
    choosing
’Continue’.
- Restart: This forces the fractal to draw itself again.

Picasso
Close Picasso
- If this item works, then it would display the 24 Bit-image directly on the ↩
    Picasso-gfxboard
from Village Tronic.

Access EGS
- This item should open a window on the EGS-Default-Screen and then should draw ↩
    the 24-Bit-fractal
in it.

Start virtual
You can choose this item only in conjunction with Julia-/Mandelbrotsets. Parameter ↩
     window No. 3
contains 4 gadgets, which let you define values like Size/Depth/3D. If you choose ↩
    this
item, the corresponding subtask switches to a virtual mode and starts the virtual ↩
    calculation directly
onto your harddisk. More information can be found in parameter window no. 3, point

                Virtual Calculation
                Export/Reflections
Well, this is the first try for an export into a raytracing format. If you have ↩
    Reflections, just test it, perhaps it
works. There will appear some requesters, which ask you for the resolution of the ↩
    object, because the program can’t use
the whole resolution, because at least ReflectionsV2.0 can’t handle more than ↩
    32767 triangles.
At the end 2 files will be saved, one file with the name *.obj, and one with the ↩
    name
*.mat.

Show Done
- Somebody complained, that he never knew exactly, when calculation was finished.
So this item is there, so you can display in the window title, how far the ↩
    calculation is
finished.
```

1.54 2.4 Menus

2.4.3 Fractalwindows

Other Menupoints:

Systemmenu

Fractalmenu

Windows

Extras
Parameter 1...
Parameter 2...
Parameter 3...
- These items open/close the parameterwindow 1/2/3. The kinds of parameters and what they mean, is explained in the corresponding chapters.

3D-Parameter 1
3D-Parameter 2
3D-Parameter 3
- The 3D-parameterwindows are opened/closed by these items. For more information about parameter refer to Chapter 2.3.7

Datenwindow
- Some types of fractals have datawindows available. If you open such a window and move over the 2D-fractal with the mousepointer, then in the datawindow the data ↩ below
the mousepointer are displayed.

Wizzard window
- Opens/Closes the Wizzardwindow

Comment window
- Opens/Closes the
Commentwindow
Formula window
- Opens/Closes the formula window.

Show Location
- Juliaset, Mandelbrotset, Bifurcation, Lyapunov-Space have a 2D-area. You can ↩ zoom in. And suddenly
you don't know exactly, where you have zoomed in, and where you now are. The area ↩ parameter in the parameterwindow 1
show it, but it's not clear enough. This item opens a window for the active ↩ fractal. In this window
all fractals of the same type are displayed. If you choose one of them, then in ↩ the 2D-window the area of the
chosen fractal is drawn as a frame.

Set Juliaparameter
- This item is only choosable in conjunction with the a mandelbrot fractal. It ↩ opens a window, in which
all available juliasets are displayed. If you choose a juliaset, then the ↩ parameter of the juliaset, mostly called c, is drawn in

```
the mandelbrotset as a cross. This cross you can move around, and so change the  ↩
    parametervalue c of the juliaset, which is drawn again
immediately. There was mentioned above, that the most interesting, that means, the ↩
     most colorful juliasets have parametervalues c, which are placed
at the edge of the mandelbrotset. But where's the edge of the mandelbrotset, if  ↩
    you only have a complex number?
With the help of the cross you know it exactly.
But pay attention: Julia- and mandelbrotsets should be of the same subtype, that  ↩
    means, they should
be drawn upon the same formula. Otherwise all said about interesting juliasets at  ↩
    the edge of the mandelbrotset
is totally nonsense.

Colormapping Window
- Opens/Closes the
                Colormapping window
                Windowtype as Backdrop/normal Window
- Eventually somebody wants to use the whole place on the screen for a fractal,  ↩
    like
most other fractal creating programs do. But if a windowborder exists, this isn't  ↩
    possible.
So you can define a window as a backdrop window. In this case, the whole window is ↩
     closed,
the border, the systemgadgets, the title removed, the window sized to the full  ↩
    screen size,
and then opened again as a backdrop-window. This can be done with the 2D/3D-window ↩
    .
But pay attention: Because you now don't have a depthgadget in the windowframe,  ↩
    you can't alter
the (depth-)position of a window. So one can dispute about the sense or nonsense  ↩
    of more than one
backdrop at the same time.
```

1.55 2.4 Menus

```
                2.4.4 Windows

Other Menupoints:

                Systemmenu

                Fractalmenu

                Fractalwindows

                Extras

                Formeleditor für Julia/Mandel

                Formeleditor für LSystem

                Formeleditor für IFS
```

Palettewindow

Palette-edit-window

Animation 1&2

CycleControl

Userwindows
Here I can refer to
chapter 2.2
. There all you should know is
said.

Perhaps one annotation: By using these menuitems, you can only open the first 4 user defined windows.
If you have more, then you have to use the Arexx-Port to open the additional windows. Then you can define
a user defined menu item, which executes your Arexx-script.

1.56 2.4 Menus

2.4.5 Extras

Other Menupoints:

Systemmenu

Fractalmenu

Fractalwindows

Windows
Help
- Shows the contents node of the online-help. If somebody wants help to a specific topic,
then he can use:

1. Menuhelp
You choose a menuitem, but don't leave the right mousebutton, so the item isn't chosen, but highlighted.
Then you press the Help-key. The operating system then reports to my program, that the user wants help for this menu item. Then my program shows the
correct page automatically.

2. Self implemented Gadgethelp
My program maintains big datalists, in which the positions and sizes of all gadgets are
stored. If you now press the Help-key, then my program scans through it's lists, searching for a gadget
below and shows the help-page for this gadget. If the mousepointer isn't placed over a gadget, then the
default-helptext for the window is shown.

Global Stop
Global Continue
- Stops calculation of all fractals, puts all tasks to sleep. Useful, if another program needs all CPU-power.
'Continue' wakes up all tasks, if they were put to sleep.

Colorcycling
- On
This item switches the colorcycling on (checked) or off (not checked).
- Upwards
This item defines, whether the colors should be cycled upwards (checked) or downwards (not checked).
Upwards means, to higher colornumbers.
- Faster/Slower
Speed of colorcycling. For colorcycling a separate task is created. According to the RKM-Libraries nobody may alter the colortables.
Because now the taskswitching takes place only ca 50 times per second (or was it 20 times?), the maximum speed of cycling is limited. Everybody, who works with 256 colors, has also to consider, that to alter 256 colors is much work for the operating system. It must recalculate the whole copperlists, link them together and display them. This takes away much CPU-time, so it slows down the system.
Btw.: The cycling task runs at a priority of 0, more axactly at GlobPri (GlobPri you can set). So when another program calculates something at a priority of 1, then colorcycling doesn't take place.

Taskpriority
- This alters the task-priority of the main task and of the colorcycling-task. All fractal-calculating tasks run at a priority of the maintask-1. Default-pri for the maintask is 0, so fractal -calculating tasks run at a priority of -1, so you can work normally on the workbench or in any other program.

Move Window...
- onto Fractalscreen / onto Parameterscreen / onto Workbench / onto Publicscreen

These items close a window and open them again on the specified screen. This can be done with every window except the 2D- and the 3D-fractalwindows. This options make sense, because the place on a screen is limited, even if it's a big screen.
This saves also memory, because a parameterwindow needs much more memory on a screen with 256 colors, than on 4-color-workbench.
You can define the default-screens of the windows by the preferences-program.

Window positions
Here you can specify the screen, on which the correspondig window should open as default.

Misc/Picture remap
This menu item belongs to 'Load picture' and determines, whether the pictures, which will be loaded, should be remapped to the current palette.

Misc/GuiFactorX
Misc/GuiFactorY
Inside ChaosPro all gadgets and windows are specified in units of the fontsize, and not in units of pixels.
There exists a global routine, which handles opening of windows containing gadgets, borders, etc. This routine
then calculates the pixelsize of all elements from the fontsize of the elements.
Due to this the whole GUI of ChaosPro is fontsensitiv. But there are many different fonts
out there. And I have choosen my favorite font, XHelvetica 11 from Martin Huttenloher (contained in MagicWB),
so the GUI looks of course best with this font. But due to this it of course may happen, that sometimes, especially with a small font,
like topaz 8, the GUI looks rather ugly.
These 2 factors now enable it to stretch the whole GUI in x- or in y-direction. This is just like
having a sizegadget on all windows, but less comfortable for the user (but more comfortable for me ;-)).
These 2 factors you can choose according to your font and according to your taste, so the GUI looks
better. After that you can save the config-file, because these 2 factors of course are placed in the config-file, so your changes
don't get lost when you leave ChaosPro.

Misc/Publicscreen
Here you can define the name of the Public screen, which can be used to place windows on it. After you have chosen this name you should save the config file, so you
don't have to specify this screen again.

Choose Screenmodes...
Choose Font...
- These items should be clear...
If they are chosen, then all opened windows are closed, the values changed, and the windows again
opened.
The minimal screendepths are:
1. Parameterscreen: 1
2. Fractalscreen: 3
3. Colorscreen: 4
- For the font all is possible. But: If a window doesn't fit onto the screen, then
you must choose a smaller font. This concerns mainly all those people, who use a resolution
of 640x200 or 640x256. Topaz 8 is almost too large.

Config load/save
- These menu items let you load/save different configurations. These files contain (almost) all internally
changeable data like screenmode, font, currently opened windows, positions of the windows
etc.

1.57 2.4 Menus

```
2.4.6 User defined Menus

For this purpose, again an ASCII-file in the directory ChaosPro/Prefs with the  ↩
    name
Menu.asc is required. This file has to be translated by the preferences-program.  ↩
    The result is
a file called Menu.prefs in ChaosPro/Prefs.

The structure of the ASCII-file is:

MENU <Menutext> <Keyboardshortcut> <Arexx-Script>
ITEM <Itemtext> <Keyboardhortcut> <Arexx-Script>
...
ITEM <Itemtext> <Keyboardshortcut> <Arexx-Script>
MENU <Menutext> <Keyboardshortcut> <Arexx-Script>
...
END <Menutext> <Keyboardshortcut> <Arexx-Script>

Well, at the lines MENU and END of course the Keyboardshortcut and the Arexx- ↩
    script
don't have sense, but must be given.
For the menutext also the constant BARLABEL may be used. It generates a separator  ↩
    bar
For the keyboardshortcut also the constant NONE may be used, if you don't want to
define a shortcut.

As an examples, it could look like this:

MENU Menu1 NONE dummy.rexx
ITEM Data B Daten.rexx
ITEM BARLABEL NONE dummy.rexx
ITEM Another C Another.rexx
MENU Menu2 NONE dummy.rexx
ITEM InOut D ChaosPro:Rexx/InOut.rexx
END BARLABEL NONE dummy.rexx

Note:
At startup the program creates a logical assign ChaosPro: to the directory, where  ↩
    the program is placed, if not already available. So you may use
a path for a rexx-script like ChaosPro:Rexx/InOut.rexx.
```

1.58 2.5 Programdirectories

```
2.5 Programdirectories

Basedirectory, from which the program refers to its various subdirectories, is  ↩
    always the logical assign 'ChaosPro:'.
If this assign at startup of the program exists, then all is ok. If not, then the  ↩
    program tries to find out, from what directory it was started (with a call
to GetProgramDir from dos.library). After that it creates itself the logical  ↩
    assign ChaosPro: to the found directory.
So normally you don't have to worry about assigns.
```

```
ChaosPro:libs/
Here all libraries, which the program needs, are placed. You may not copy these ↩
    libraries
to LIBS:, because only my program needs it, they aren't documented and, by the way ↩
    , they
aren't really libraries. Due to this, the whole program is very easy to deinstall. ↩
     Simply
delete the main directory, if you don't like the program...

ChaosPro:Guides/
Here the documentation of the program is placed.

ChaosPro:Prefs/
All settings of the program are placed in this directory.

ChaosPro:Palette/
ChaosPro needs at least one palette in this directory. Otherwise it refuses to  ↩
    work
and brings up an error requester. At startup it scans this directory, examines all ↩
     files
in it, and extracts all colorchunks of the files. So its possible, to place whole  ↩
    pictures
into this directory. It then scans through the file and only takes the color
chunk of it.

ChaosPro:Catalogs/
In this directory the catalogs for other countries are placed. Because I only  ↩
    speak german and
english (and english not very good...), here only two catalogs are made by me.  ↩
    Perhaps
some other people would like to translate the catalogs?

ChaosPro:FractPic/

At startup of the program this directory is scaned. All files, which contain a  ↩
    chunk
describing a fractal, are automatically loaded into the program. If a fractal  ↩
    needs a user defined formula, then it loads it,
too, if it doesn't already exist.

ChaosPro:Anims/
ChaosPro:AnimData/
ChaosPro tries to load/save animations or animationdatas from these directories in ↩
     the first
place.

ChaosPro:Formula/
This directory is scanned at startup, too. All user defined formulas, which aren't ↩
     already
in memory, are loaded and can be used during runtime.
```

1.59 2.6 Preferencesprogram

2.6 Preferencesprogram

In order to translate specific files there exists external preferences program. It offers
the following options:

Compile Userwindows

User defined windows
are defined in an ASCII-file. This ASCII-file
must be translated into a format, which the mainprogram can more easy handle. You simply have to click
onto this gadget. Then the ASCII-file ChaosPro/ Prefs/ Windows.asc is scanned and the
file ChaosPro/ Prefs/ Windows.prefs is created.

Compile Usermenus

User defined menus
are also defined in an ASCII-file, which must
be translated into another format. This is handled by this gadget. It creates from the input-file ChaosPro/ Prefs/ Menu.asc
the output-file ChaosPro/ Prefs/ Menu.prefs.

The Online-Help
Because the program can run on any normal screen, it was necessary, to adjust the
online-help to run on any screen in any resolution in any font. Other programs only
have a help-system, which looks rather good on a screen with 640 pixels horizontally.
But then some people have a screen with 1024 pixel horizontally, and then the help-system
is awful. What happens, if the user would like to use another font for the help?
It would look terrible. Suddenly all lines have different widths, if you use a proportional
font. Due to this, the guide-file is also translated by the preferences-program, so it
looks good on any screen in any font.

- GuideWidth
Here you define the width in screenpixel, the help-lines should be. Because the windowborder also needs
some pixels, you normally have to subtract about 40 pixels of the with of the
screen, the help-system should run on.

- Language
Here you define the language of the help-system. This is independant of the AmigaOS
locale-system. Included are only 2 languages: german and english. Perhaps some other people would like
to translate the online-help to other languages?

- Build Guide
If you click onto this gadget, then in the directory ChaosPro/Guides the guide-file ChaosPro.guide is created. The original
files aren't modified. ChaosPro.guide is a normal AmigaGuide-file, which you can read with MultiView, Hyper

or AmigaGuide, but which is converted to the right format. This operation can take a
long time (on my Amiga 4000/040 about 60 seconds), because 1. the guide-file is quite long and 2.
I didn't care, whether it's slow or fast, because 3. I think, that you'll use this option
not very often.
Dependent on the font it may happen, that a mysterious requester comes up reporting the error
"Failed to create a line". Here you only can click onto 'OK' and this you should
do.
In order to change the font of the guide-file or to change some explanations, you can
of course change the file ChaosPro.guide. But this is somehow not intelligent, because then
you don't have the right format and all changes, you made, are destroyed, when you again
click onto 'Build Guide'. So if you want to change something, you have to change deutsch.guide
or english.guide. These files are also normal guide-files for AmigaGuide, Hyper or MultiView,
but they aren't in the right format. All lines have different lenghts. In these files you can
change the @FONT-directive and set another font and size. If you want to change some
text, then you have to pay attention, because a paragraph is finished with a line, which
contains less than 76 characters. If you create a line with equal to or more than 76 characters,
then this line will be concatenated with the following, forming a paragraph, these lines with
the next, etc., until a line is encoutered with less than 76 characters, which stops
the paragraph. If a word is too long, it perhaps don't look too good in the guide. Due to this,
you can define, where a long word can be separated. This you define with the backslash-character left
to the backspace-key at the right top of the keyboard. So you only have to insert this character at the
correct places. The program then eventually separates the word at this position, inserting a '-'-character
or it simply removes the backslash.

1.60 2.7 Troubleshooting

2.7 Troubleshooting

1. Problem
Sometimes the system hangs, when I try to use the Online-help.
Solution:
No solution. I don't have an explanation of this behaviour. But because the AmigaGuide isn't
totally bugfree, I think, this is a failure of the AmigaGuide-system, which is really

not bugfree.

2. Problem
How can I set the size and position of the AmigaGuide-window?
Solution:
By choosing the menuitem 'save settings'...

3. Problem
If the Help-Key is pressed, no AmigaGuide window appears, so the Online-help doesn ↩
't work.
Solution:
1. Perhaps the AmigaGuide-System isn't installed correctly. In this case you ↩
should get the
complete official AmigaGuide-distribution. Then you should install it.

2. Perhaps ChaosPro.guide or ChaosPro.Topics isn't available. In this case you ↩
should start
CPPrefs and click onto 'Build Guide'. More information about this you will find
in the chapter
Preferences-Program
3. There exists a tooltype, which can be used to disable the ↩
Online-help, thus saving
memory. It is called 'NO_AGUIDE' and, if specified, prevents the program from ↩
initializing
the Online-help.

4. Problem
ChaosPro crashes at startup.
Solution:
Well, one possibility, which was true in about 60% of all cases, is, that the user
has installed FastMathV40.5. This version has a serious bug, so you should upgrade
to FastMathV40.6...

1.61 2.8 Others worth mentioning

2.8 Others worth mentioning

How do you input numbers? Of course with integergadgets. But what's with ↩
floatingpoint-numbers?
Unfortunately gadgets for this don't exist in the system. So I was forced to write ↩
a
Hook-function of my own, in order to make a float-gadget out of a string-gadget. ↩
In this
float-gadget all senseless keypresses are filtered out. Some other key- ↩
kombinations like RAmiga+X,
in order to clear the inputfield, make actions, which I think, they should do. ↩
RAmiga+X
writes into the field the number '+0.0'.
In order to alter the sign of the number, you only have to press the key '+' or ↩
'-' at any
place in the field. The sign at the first place changes immediately.
In order to set the decimal-point to another position, you simply have to press ↩
the '.'-key

at the desired place. The eventually already existing decimal-point is cleared and ↩
    set
to the new position.
Numbers in exponential-expression aren't possible in the current version of the
Hook-routine.

Everybody, who has already used the program, will have noticed, that the active ↩
    entries in
the PicTask-window are changing sometimes, if the user activates another window. ↩
    This is of
course not random, it's made by the program. The active window determines the ↩
    active entry in the
task-listview. Whenever you activate another window, the program searches for the
task, the window belongs to, and declares this task as the active one. ↩
    Additionally, it
scans through the whole menu and actualizes the items, so it disables some, ↩
    enables others and makes
checkmarks according to the active task.
If you don't know any more, what fractaltype you are currently examining or to ↩
    what
task the window belongs, then you should have a look at the screentitle. There the
name of the fractal and the fractaltype of the task is displayed, which belongs
to the window.

I've tried to write this program style-guide conform. Due to this my program isn't ↩
    the fastest fractal-generating-program.
Especially the owners of Mand2000 from CygnusSoft will notice, that my program isn ↩
    't very
fast while scaling the windows.

Some people want to open some windows at startup automatically, or that something ↩
    other happens
immediately at startup. This possibility is offered through the AREXX-Port. At ↩
    startup
the rexx-script ChaosPro/Rexx/ChaosProInit is executed. There you can execute all
commands you wish.

Well, almost all programs want a logical assign for their work. My program wants ↩
    something
like this, too. But I've applied another method: The program searched at startup ↩
    for the
logical assign 'ChaosPro:'. If this assign is available, then it searches for the ↩
    various
subdirectories, for example ChaosPro:Prefs, ChaosPro:Palette, ChaosPro/Formula etc ↩
    ., in order to
get its files.
But if this assign isn't available, then it creates it itself and removes it at ↩
    the end.
This means, that you may use at runtime, as example in your Rexx-scripts, the ↩
    assign 'ChaosPro:'.
It's available in every case.

1.62 2.9 Tooltypes

2.9 Tooltypes

The program currently supports the following tooltypes:

NOJOYSTICK
This command disables moving and zooming around with a joystick in port 2. This was made,
because it's possible, that somebody has a dongle in this port, which might cause strange things,
if port 2 is accessed. So if you use a dongle (perhaps the REAL3D-dongle), then specify this
tooltype.

CHUNKYMODE
This command specifies the routine to use for scaling the fractal, if you do a doubleclick.
Normally this is done in the following way: The whole content of the window is read with
ReadPixelArray8. After that the buffer, which contains the values, is scaled by a routine of
my own. After that this buffer is converted to planeformat with a ChunkyToPlanar- routine of
my own. After that ClipBlit is used to copy the planedata into the window. Now consider, that
somebody has a Gfxboard and a the program runs on a screen with a chunkymode. Then of course all
works, but: After the buffer is scaled, my program converts it into the planeformat, then I execute
ClipBlit, which is patched and internally converts the planedata back to the chunkymode of the gfxboard...
If you specify CHUNKYMODE, the program doesn't use a ChunkyToPlanar and ClipBlit, but a WritePixelArray8, and the gfxboard
can take the values as they are, it needn't convert them. Please note: The whole program never directly accesses any planes of
any window on the screen. If all programs would be so, then fantastic fast gfxboard-drivers could be
written.

COLORWHEEL
This tooltype specifies, whether the colorwheel should be shown for the palette- editing.
Because it needs several colors to look like a "colorwheel", half of the number of colors
on the screen are used for it. If you want to use all of the colors on the screen for the palette-colors,
then don't specify COLORWHEEL.

BUILTIN
If specified, then the builtin language (english) will be used. Otherwise the language specified
by the locale-system. Only useful for me to some routines...

BACKFILL
If specified, then the window is filled with a raster before all gadgets are added.
Well, it's up to you to decide, whether you like it or not...

```
PGA_NEWLOOK
If specified, the proportional-gadgets get the 'new look'. GadTools doesn't ↩
    support it.
So this bit is set by hand. Well, it works, but it's an undocumented feature, ↩
    which don't
need to work. Several authors use this bit and none of them encountered any ↩
    problems.
(Normally it's not allowed to alter ANY bits in a GadTools-Gadget...)

NO_EGS
If not specified, the EGS-System, if installed correctly, can be accessed by a ↩
    menuitem. This item draws the actual fractal into
a window on the EGS-Default-Screen.

NO_AGUIDE
If not specified, the amigaguide.library is opened. This will add a ↩
    kontextsensitive
online-help to the program. If you don't want it and want to save memory, then ↩
    specify
it.

NO_REXX
If not specified, the Arexx-Port of the program is initialized and the rexxsyslib. ↩
    library is opened, so
the Arexx-Interface is available. But perhaps you don't use it, then why should it ↩
     be initialized and
consume memory? In this case specify this tooltype.

PICTURES=<Dir>
This tooltype specifies the directory for the pictures. Default is 'ChaosPro: ↩
    Pictures'. Because everybody
has another favorite place for the images, you can specify this.

EXPORT=<Dir>
This tooltype specifies the directory for the exported objects in the Reflections ↩
    file format.
Default is 'Szenen:'.

PALETTES=<Dir>
This tooltype specifies the directory for the palettes. Default is 'ChaosPro: ↩
    Palette'.
If your palettes are in another directory, you may change the directory. Please ↩
    note: ChaosPro will
still on startup load only the palettes present in 'ChaosPro:Palette', this ↩
    tooltype specifies only the
default directory for the palette filerequester.

VIRTUAL=<Dir>
This tooltype sets the directory, in which the temporary files for the virtual ↩
    buffer will
be placed. This should be a directory on a partition, which has much place free.
Default is 'ChaosPro:'.

ANIMS=<Dir>
This tooltype specifies the default directory for the animation filerequester. ↩
    Default
is 'ChaosPro:Anims'.
```

```
STARTPRI=<Priority>
This tooltype specifies the start priority of ChaosPro. ChaosPro itself will run ↩
    with this priority, every subprocess will have 'StartPri-1'. Default
is 'StartPri=0'

AUTOSAVE
If specified, ChaosPro will automatically execute the menu item 'Save Config' on ↩
    program termination.
The file 'ChaosPro:Prefs/ChaosPro.config' will be saved, so at the next startup
the windows automatically will open at the old place.

DEBUG
If you specify this tooltype, ChaosPro will inform you about the progress in the ↩
    initialization procedure.
This is usefull especially when you have trouble starting ChaosPro, because then ↩
    you will approximately
know, where ChaosPro has trouble, so you can fix these troubles, or you can at  ↩
    least let me know, where
ChaosPro has trouble. It's of no use to write me, that ChaosPro doesn't start. I ↩
    need some more information.
So if you contact me about problems on startup, then please write me the output, ↩
    which ChaosPro makes, when
you specify this tooltype.

EHB_HAM
If you set this tooltype, then the screenmode requester displays the EHB and HAM ↩
    modes, too.
ChaosPro doesn't support these modes, but it works. Until now I didn't find out ↩
    the sense of the tooltype, I just implemented
it due to a suggestion of a user...
```

1.63 2.10 Legal Stuff

```
2.10 Legal Stuff

While developing this program, bugs in it crashed my harddisk a few times. So be ↩
    warned.
There are for sure bugs in the program which can cause bad things...

So:
                  No Warranty

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW.
EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDER AND/OR OTHER PARTIES
PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR  ↩
    IMPLIED,
INCLUDING, BUT NOT LIMITED TO,THE IMPLIED WARRANTIES OF MERCHANTABILITY AND  ↩
    FITNESS FOR A
PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE  ↩
    PROGRAM
IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL  ↩
    NECESSARY
SERVICING, REPAIR OR CORRECTION.
```

```
IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY  ↩
    COPYRIGHT
HOLDER, OR ANY OTHER PARTY WHO MAY REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE
LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL,SPECIAL,INCIDENTAL OR  ↩
    CONSEQUENTIAL
DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT
LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY  ↩
    YOU OR THIRD
PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN  IF  ↩
     SUCH HOLDER OR OTHER PARTY
HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.


ChaosPro 1995 Martin Pfingstl

ChaosPro is Public Domain.
```

1.64 2.11 Searching for...

```
2.11 Searching for...

I'm searching for:

1. Translater of the catalogfiles: Mail me, then you get the .cd and .ct-files.

2. If you have suggestions to improve the documentation, you are welcome to do so.
Especially if you want to correct my bad english...

3. If you have calculated some nice images, mail me the data files of them.
```

1.65 2.12 About the Speed...

```
2.12 About the Speed of ChaosPro

Well, and now I have to say something about the speed of ChaosPro. Many people  ↩
    seem to love
ChaosPro, they almost only complained about the speed.

Now I want to say the following:

ChaosPro wasn't written to claim, that it's the 'fastest fractal program ever'.  ↩
    Other programs are faster.
But on the other side I didn't write 'slow' routines. ChaosPro isn't that slow,  ↩
    than you may think after
the first speed comparison between other programs.

What some people didn't realize exactly is, that the choice of the parameters has
enormous influence on the speed. Just an example: Outside coloring=CPM. What  ↩
    happens,
if you compare the speed? The other programs just calculate an integer value for  ↩
    every
pixel on the screen using the escape time algorithm (that's the one you know...)
```

```
After that (if multiple pass is selected), they examine little squares, whether ↩
    the 4 corners
all have the same value. If yes, then the whole square is set to the value of the ↩
    4 corners.
This of course speeds up the calculation process quite a lot. These people now ↩
    think, that
it makes no difference, if they just choose CPM in ChaosPro. Well, that's wrong.
If you use this mode, then the program calculates a float number for every pixel ↩
    on the screen.
Of course it's very unlikely, that the 4 corners of a little square have all the ↩
    same float
value.

Inspite of this, ChaosPro was really slower than other programs, even if you ↩
    selected only
the standard parameters (but not that much, as people thought...). Because I don't ↩
     want people
to complain about the speed, I speeded up the routines. These speedups affect the
iteration functions, so you will notice it, when you zoom in. Inspite of this, the
standard mandelbrot will still need more time to finish than with other fractal
generators. This is due to the drawing routines I use, i.e. due to WritePixelLine8 ↩
     and
ClipBlit. Other programs just 'poke' directly into the bitplanes, thus making it ↩
    not
totally systemconform. I refuse to write such hack-routines. If you know how to ↩
    speed
up drawing routines, then let me know...
```

1.66 2.13 Changes since V1.0

```
2.13 Changes since V1.0

I want to split the changes into 2 parts:
First part: Changes, which you normally will notice,. That's the more interesting
stuff...

Second part: Changes, which you normally won't notice, for example bugfixes, ↩
    changed behaviour,
slight corrections or something similar...

a) Changes, which you will notice

- ChaosPro speeded up (Black_Hole, resolution 640x495, 1 Pass, etc. in V1.0: ca 8 ↩
    min,
now: ca. 6 min)

- ChaosPro now is able to read Mand2000, Mandelmania and compatible fractal data
files.

- Virtual Julia- and Mandelbrotfractals: Parameterwindow 3 lets you specify some ↩
    values, then choose
menu item 'Start virtual'

- Colormapping window added
```

```
- graphical representation of colormapping added.

- Wizzardwindow added

- Commentwindow for every fractal added

- FractInt-Palettes (these *.map-files) can now be loaded and used with ChaosPro
.

- 3 different Dockwindows with gadgets for some actions.

- New feature, new tooltype: AUTOSAVE. If you specify this, then ChaosPro will
write the file 'ChaosPro.config' into the ChaosPro/Prefs/ drawer, as if the user ↩
    has choosed the menu item
'Save config'. So the program at the next startup will appear in almost the same ↩
    look as you left
it.

- Old formula window changed. Now you only can choose the formula for the ↩
    calculation, if you want to change a formula, you have to open
another window.

- Formula editor now unterstands logical operators: '==' '!=' '>' '<' '>=' '<=' 'T ↩
    '
'F' '!'

- New mode for the wizzard window: If choosed, then the graph of the heights ↩
    transformation function will be shown.

- Formula editor improved (each formula has several parts now)

- ChaosPro now is able to load pictures, if you have the datatypes.library ↩
    installed
.

- DEM (Distance Estimator Method) built in. This calculates an estimation for the ↩
    distance of a point to the border of the
set. This mode is slow, because one has to calculate actually 2 iteration ↩
    functions, not only z^2+c, but also
f=2*z*f+1 (derivation of z^2+c)

- New fractal types: IFS/LSystem/Brown/Diffusion, where each IFS and LSystem have ↩
    an additional formula
editor.

- Plasma now may be calculated in 3D, which leads to mountains and nice landscapes
.

- Lightintensity-Gadget built in, which determines the light intensity, so you can ↩
     calculate
an animation, where the light slowly is switched on.

- Reflections-Export added


------------------------------------------------------------------------------
b) Changes, which you normally won't notice
```

```
------------------------------------------------------------------------------

- Closing a window and reopening on another screen didn't work for windows of  ↩
    fractals, if the
fractal itself wasn't calculated

- When changing the screenmode all calculated fractals (i.e. their windows) got  ↩
    the
default size, but the program itself thought, that they had the same size as  ↩
    before --> fixed, now the window
is opened in the same size as before...

- If an AnimData file was saved, and the corresponding animation is a 24 Bit Anim, ↩
     then the
program complained about a malformed AnimData-file. Planes was 9, program checked  ↩
    for
>8...

- CPPrefs-Window could be too big, if the font was too large. Because  ↩
    WA_AutoAdjust wasn't specified, it
didn't open, so ChaosPro couldn't be run.

- Stacksizecheck was wrong, if the program was started from the CLI.

- If ChaosPro.Topics for any reason has a size of 0 Bytes, then ChaosPro  ↩
    complained
about too few memory, because AllocMem(0,MEMF_CLEAR) always returns 0.

- If a window was too big for the screen, then ChaosPro should have taken the  ↩
    small font, set with
CPPrefs. But ChaosPro calculated all sizes and positions, as if it would take this ↩
     small font and
then took the big font...

- Tooltypes added: Anims, Pictures, Palettes, Virtual, StartPri, to specify these
directories.

- Maximal number of iterations in the inside/outside area from the datawindow  ↩
    removed
.

- Formeltextgadget wasn't adhered to the listview in the window.

- New colormapping function added: Sqrt

- Taskpriority of ChaosPro now is shown in the SystemInfo-Requester.

- If opening a screen fails, then ChaosPro shows the exact error obtained from the
operating system.

- ChaosPro just cleared the first 65535 elements in the buffer, if you changed the
maximum number of iterations.

- If you changed the iteration depth, then the fractal was just drawn again and  ↩
    not calculated
again.
```

```
- Formelparser didn't like '-'.

- Abort condition 3 removed. Now used in conjunction with the formula editor
.

- New Tooltype: DEBUG

- DEM Gadget in the datawindow added

- Eliminiation algorithm had a bug.

- PicTask-Window now isn't refreshed every time a new fractal gets active. Only ↩
    the gadgets, which
have changed, are refreshed. This avoids this nasty flickering.

- Palette editing didn't always work as expected. Grey scale palettes were wrong ↩
    and 'spread'
didn't work with grey scale palettes.

- If you save a picture, then the CRNG-Chunk is saved. Now I only need a program, ↩
    which
pays attention to this chunk (CRNG=Colorcycling RaNGe)

- Moving the fractal past the left border didn't work.

- Function imag(...) was implemented wrong.

- Formula editor didn't like spaces
```

1.67 Some Cookies (sorry, couldn't resist)

```
... A bus station is where the bus stops.
    A train station is where the train stops.
    On my desk there is a workstation...

   "All I know is what I see on the monitors."


Computing Definition:
Microsecond - Amount of time needed for a program to bomb.

  1st Law Economists: For every economist there exists an equal and
      opposite economist.
  2nd Law Economists: They're both always wrong!

Sega brings you its new Baby....CBM brings you the MOTHER..CD32

           "Research shows that no-one ever went blind from
                  looking on the bright side of life"


"I really wish I'd listened to what my mother told me when I was young."
"Why, what did she tell you?"
"I don't know. I didn't listen."
```

```
Reality is just a big simulation -- And it's still in beta-testing !

Living on Earth may be expensive, but it includes an
annual free trip around the Sun.

>>> Life starts at '020 ... fun at '030 ... impotence at '86 <<<

keyboard not connecte+d -- press F1 to continue

WindowsError:010 Reserved for future mistakes

WindowsError:011 Hard error. Are you sitting?

America has Bill Clinton, Steve Wonder, Bob Hope & Johnny Cash
we have Helmut Kohl, no Wonder, no Hope & no Cash !

 MS-DOS is the worst text adventure game I have ever played: poor vocabulary,
                       weak parser and a boring storyline.

WindowsError:003 Dynamic linking error. Your mistake is now in every
                 file.

Windows NT: From the makers of Windows 3.0!

GEOS ON C64 IS MUCH BETTER THAN WINDOWS ON PC

* Englishtraining for runaways:
*
* Don't worry, eat Chappy

Nuclear clock stands at T - 5 minutes

Don't marry be happy

WindowsError:014 Nonexistant error.  This cannot really be happening.
```

1.68 2.14 Many Thanks and Greetings to...

```
2.14 Many Thanks and Greetings to...

Well, let me thank all the people, who have supported me and my work on ChaosPro
:

Olaf Krolzig: A betatester since the beginning. He has suggested many features,  ↩
    constantly reported many
bugs.

Lutz Uhlmann: This was my first betatester. At least it was the first one of the
betatesters, who was able to start ChaosPro. He has suggested to implement a  ↩
    screenmode- and
font-requester

Kay Gehrke: He constantly reported bugs concerning the Picasso-graphics card. So I ↩
     finally managed it to write
```

a system conform program.

Manfred Ambros: He tested all stuff. He also tested the documentation, reported wrong links and some spelling
mistakes...

Jake and Mac Melon: These two betatesters persuaded me to rewrite the animation system, to add the spline-interpolation between key frames, to enhance the 3D- transformations,
etc...

Roberto Patriarca: He is responsible for the italian translation of the catalog- file. He also
offered his help to translate the whole documentation to his native language. I dissuaded him from doing so.
I think, after the translation of the documentation he wouldn't want to hear the word 'ChaosPro' again. I think, a friend, a user of ChaosPro,
is better than an italian documentation...

Bruce Dawson: He was written Mand2000 (Cygnussoft) with its genious user interface .
If you would have suggested me to write a fractal generating program, which calculates the fractals into windows and is able to calculate more fractals
at the same time, then I would have asked, why I should do so, because I thought, it doesn't make much sense.
Bruce Dawson with Mand2000 proved, that this in fact makes much sense. So Mand2000 is/was the pattern for
ChaosPro. Many thanks to Bruce Dawson!

1.69 Index

III. Index

```
   -  size
   -  start/abort
   -
                window
                backdropwindow
Bifurcation:
   -  A
   -  cykluslength
   -  data window
   -  formula
   -  iterations (data)
   -  iterations (parm)
   -  parameterwindow 1
   -
                theory
                   -  values of variables
   -  variable to use
   -  variables
boxzoom

calculate picture
clear task
colorcycling
continue calculation global
continue calculation local

datawindow
delete picture
duplicate picture
dynamic system:
   -  area
   -  drawmode
   -  parameterwindow 1
   -  parameterwindow 2
   -  speed
   -  start
   -  systemtype
   -
                theory
                   -  timeparameter
   -  viewangles

formula editor:
   -  add formula
   -  edit formula
   -  formula in/out
fractal pictures
fractal tasks
fractaltypes
fractalwindow

juliaset:
   -  abort conditions
   -  area
   -  bailin
   -  bailout
   -  biomorphy
```

- circle inversion
- datawindow
- decomposition
- drawpasses
- formula
- inside coloring
- iterations
- outside coloring
- parameter
- parameterwindow 1
- parameterwindow 2
- parameterwindow 3
-

theory
lyapunov-space:

- area
- chaoscolor
- data
- drawpasses
- formula
- iterations
- minimum of exponent
- parameter
- sequence
- stabilization
- start
-

theory
mandelbrotset:

- abort conditions
- area
- bailin
- bailout
- biomorphy
- circle inversion
- datawindow
- decomposition
- drawpasses
- formula
- inside coloring
- iterations
- outside coloring
- parameter
- parameterwindow 1
- parameterwindow 2
- parameterwindow 3

move fractal

palettes/edit palettes:

- actions
- areas
- colorcycling
- colornumber
- colors
- copy
- delete
- duplicate

```
   -  edit
   -  editwindows
   -  HSV
   -  in/out
   -  invert
   -  name
   -  palettewindow
   -  RGB
picturename
place window on another screen
plasma:
   -  granulation
   -  parameter
   -  proportion
   -  randomseed
   -
                theory
                proportion
preview

quit

recalculate fractal
redo

save data
save picture as ILBM
set juliaparameter
set values to default
show done
show help
show location
stop calculation global
stop calculation local
systeminfo

taskpriority
theory to:
   -
                bifurcation
                   -
                dynamic system
                   -
                juliaset
                   -
                mandelbrotset
                   -
                lyapunov-space
                   -
                plasma
                undo
user defined windows

zoom
```