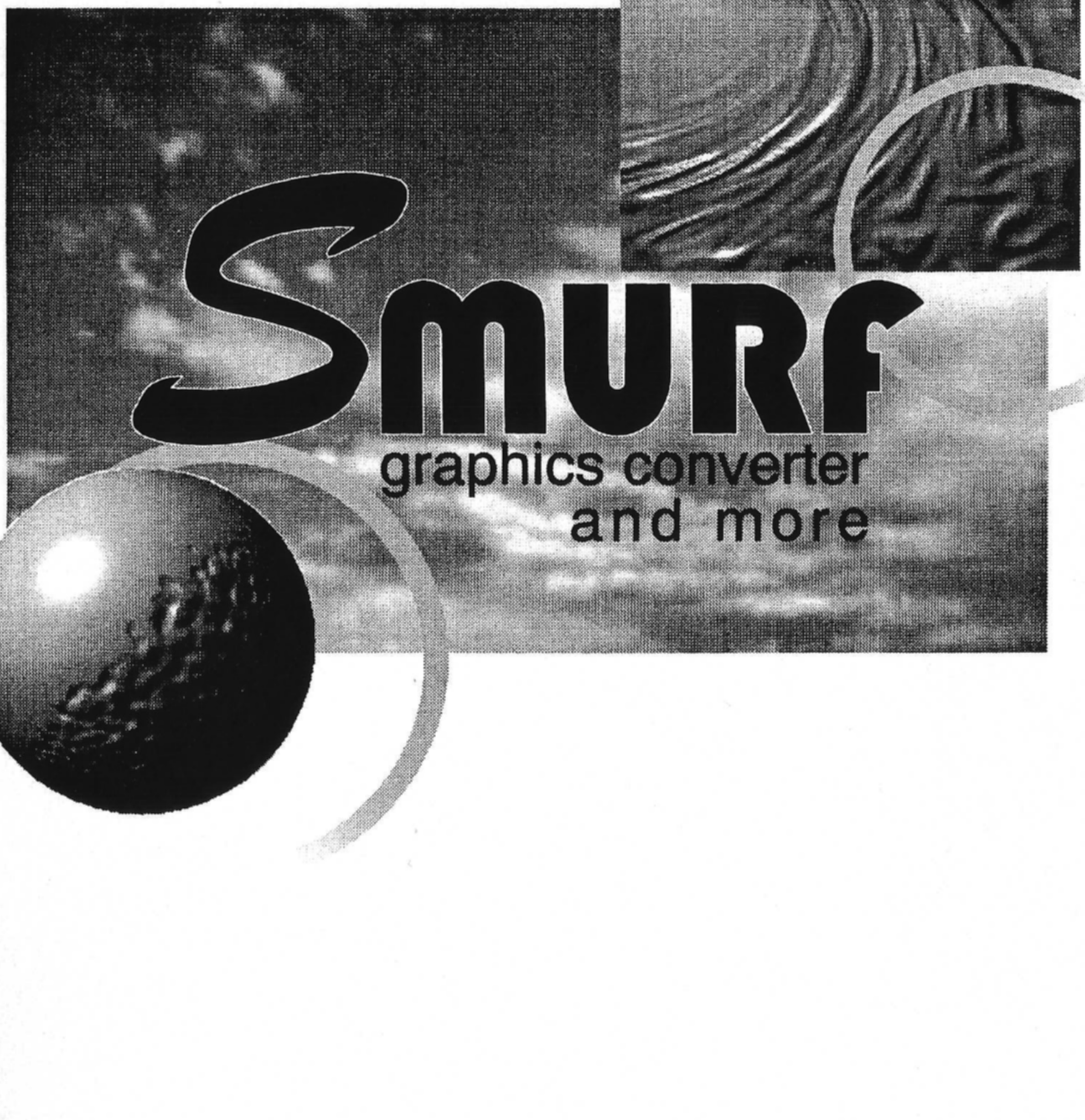


Therapy Seriouz Software
Olaf Piesche, Christian Eyrich GbR
Nordistrasse 28
D-90455 Nuernberg

e-mail: therapy@therapy-seriouz.de

THERAPY
SERIOUZ SOFTWARE

The program, the manual as well as the figures were done with largest attention. Nevertheless errors cannot be excluded. Therapy Seriouz Software is not liable for injury or detriment which is caused due to errors in the manual or in the program.
Modifications of in the manual described functions or procedures are subject to change without notice.

IMPRINT

2. Edition

Therapy Seriouz Software
Olaf Piesche, Christian Eyrich GbR

EDITORSHIP:
Christian Eyrich
PICTURES:
(X)IMG-Snapshot
CODE:
Olaf Piesche
PRINT:
Brother HL-760, Xerox DocuTech
DESIGN:
What?

Index

Who is Smurf?

"Is it far to go, Papa Smurf?"

The Smurfs are those tiny blue beings with the white caps, who live in Smurftown and actually could be very happy, if there wasn't the mean wizard Gargamel and his cat Azrael.
But actually this has nothing to do with our program, except for the name, and the Gargamel. The question why is today not being answered, just as the question, for what the heck 42 is the answer.

On this point the german manual has
a very funny version of the poem
"Der Erlkönig", called "Der Ditherkönig".

But we fear, a simple translation can't keep
the brilliance of the work. To this we don't
know if "Der Erlkönig" is known out of Germany.

So here you only find nothing else than:

"On this point the german manual has
a very funny version of the poem
"Der Erlkönig", called "Der Ditherkönig"

But we fear, a simple translation can't keep
the brilliance of the work. To this we don't
know if "Der Erlkönig" is known out of Germany.

So here you only find nothing else than:

"On this point the german manual has
a very funny version of the poem

a.s.o.

OK, once again.

1 Prologue

Smurf is a modular graphics tool for all ATARI Computers from to ST upwards. Smurf needs at least 512 KB of free memory and a minimum screen resolution of 640x400 pixels. Nevertheless, we recommend 1 MB free memory and a harddisk (image files are not getting smaller and floppy disk drives not faster, even with Smurf ...) and a graphic accelerator like NVDI.
Graphics cards are supported, which means, there should be as less or as much problems with any other GEM application.
Smurf has been tested with TOS, MultiTOS, N.AES, MagiC, MagiCMac and MagiC PC on a 520ST, several TTs, Hades, PowerMacintoshs, Pentiums and, of course, several Falcon030s, in various system configurations.

Now, what's it all about?
A big problem was, even or especially on ATARI computers, to get the huge amount of different image file formats handled properly when working with different graphics or wordprocessing applications. And this not only from Doodle to Degas and NeoChrome to STAD - the ancient times of isolation are gone. So there are very frequent confrontations with the different graphics formats from different computer platforms. And that's what Smurf was basically made for.
There are 67 different modules for import and 15 for export, this should cover most of the important (and even some of the unimportant :-)) file formats.
Because Smurf is a modular program, concerning import and export as well as the image processing functions, expandability and flexibility is guaranteed. We're constantly working on the improvement of the modules and of course on the development of new modules. Most of them are going to be freeware and will be available on our website.

1.1 general information on this manual

What is this manual, except of a contradiction to our statement, Smurf was an intuitively operateable program?
Manuals can be used for many things. Apart of all those like fly-swatter, fan or support for loose desks, this manual is made to be a leader. When you first start Smurf, it is going to explain, what the different menus and dialogues mean. And with the further working with Smurf, you can read more about hidden and advanced functionality.

And even if you can handle Smurf, there is lot of background information about graphics and graphics programs in this manual.

During all the time, the manual has been written, the main goal was to create a readable and easily understandable manual. I hope I was successful (so do I - the translator), even if this is in contradiction to my program code.
For better orientation we used **this font** for all buttons, texts and menu titles. It's not quite smooth, but that is no mistake - it's because it displays things you can see on your monitor.

General information on the english translation:
I'm not what someone would call the prototype englishman - as you will see when reading this manual. I want to excuse for all mistakes I've made in the translation of the german original, they've mostly not been made on purpose rather than accidently. If there should be understanding problems because of some unreadable phrases or word combinations, please send me an e-mail (olaf@therapy-seriouz.de). I'll explain whatever it is and take the appropriate corrections in the manual's text (tips are welcome).
Thank you and have a nice day.
The translator

Of course this manual is not the only help for you. There is also the

1.2 ST-Guide - Help

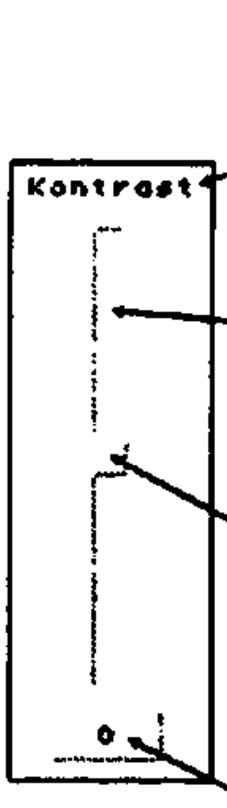
Parts of the manual, that explain the most important dialogues, can be found as context sensitive online help files for Holger Weets' famous hypertext system ST-Guide.
This online help is called by the HELP key and always refers to the top window on your screen. For online help, ST-Guide has to be running as a desk accessory or application. Pressing the HELP-Key opens the appropriate help file.

If the installation has been finished properly, that should be everything you need to do. The installation program copies all hypertext files into the right directory on your harddisk, or assigns a new directory to the ST-Guide configuration file, by request.

1.3 general informations on the GUI

1.3.1 Sliders

The sliders can basically be handled as you are used to handle a slider. That means, there is a kind of button that can be moved upwards and downwards and that adjusts something. There are a few not so common things about Smurf's sliders. Let's take a look at a standard slider in Smurf.



caption: *tells you, what this slider adjusts, anyway.*

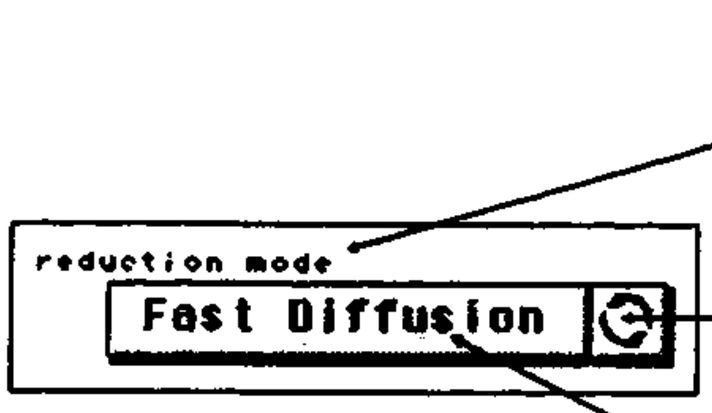
The slider can be moved inside of this box . A left klick on the box moves the slider one pixel down or up.

Slide button: *klick and drag.*

Display: *displays the current slider value (depends on the function of the slider). Left klick opens a small popup to directly type in a value.*

1.3.2 Popup menus

They're called so, because they pop up on a klick. Smurf popups always look the same and can be used the same way.



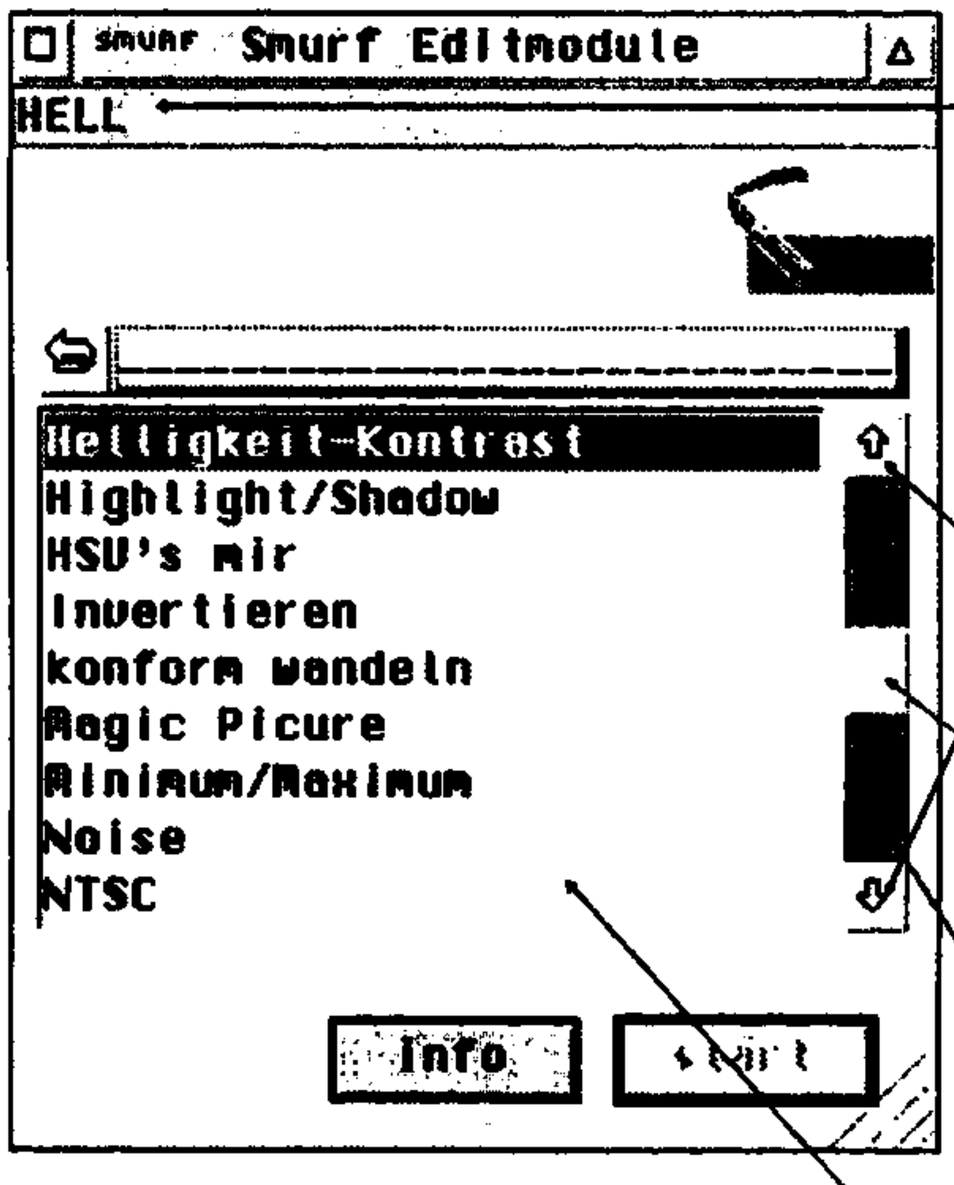
Caption: *Shows you what can be configured by the popup menu.*

Cyclebutton: *Cycles through the entries of the popup menu. Left klick:forward, right klick: backward.*

Popupbutton: *Opens the menu in which you can choose one out of several entries.*

1.3.3 List fields

They're normally for choosing modules or things like that. List fields also always look alike in Smurf:



Autolocator: *If the window is topped you can directly access one of the entries by typing it in. Backspace deletes one letter, Escpape all. HOME / SHIFT-HOME jumps to the beginning/end of the list.*

Arrow buttons: *Are used to scroll up or down one entry in the list.*

Slider: *Klick and drag to scroll realtime.*

Slider background: *Left klick above or below the slider makes the list jump one page up or down.*

List: *Left klick selects an entry. The selector can be moved up or down with the cursor keys.*

1.4 a few words about plugins

Smurfs plugins subdivide in three different classes.
First of all the ones who represent fixed internal functions of Smurf and have been put into plugins for memory saving and flexibility purposes. At the moment these are the Multiconverter and the print plugin. Although both are plugins they're not listed in the plugin menu but have their own menu entries "Multiconversion" and "Print" in the "File" menu.

The second class of plugins adds new functions into Smurf that are not necessary or standard and can, not have to be there. These can be found in the plugin menu.

The third class replaces or enhances Smurfs program internal functionalities - these plugins replace or add new dialogues and functions, change Smurfs behaviour, add new buttons in Popup menus and similar things. These plugins will not be listed in a menu, you will never see them - except for the results they have on Smurfs internal functions.

All plugins have in common that they are located in the **MODULES\PLUGIN** subdirectory in Smurf's folder.

Active Plugins have the ".PLG" file extension. Deactivated plugins have ".PLX". Renaming a file from .PLG to .PLX (or vice versa) and restarting Smurf will deactivate the plugin (or activate it).

Normally you don't get too much to do with the file names, only when it's about the printing plugins. Because there are two of them. One with its own window dialogue, which runs without the system extension WDIALOG. This is the one that will be activated after the Smurf installation, with the filename **PRINT.PLG**.
The other printing plugin needs the WDIALOG system extension and uses the printing dialogues that are implemented there. The filename of this plugin is **PRINT_WD.PLG**.
So if you want to switch between the two plugins ... now you know how.

2 Installation

"I'm the Installer, Ma'am ..."

It happened. In spite of our earlier statements Smurf is now delivered with an installation program. It's quite easy to use, but we're going to explain it anyways.

To install Smurf simply start **INSTALL.PRG** from the Disk. A dialogue will open, and you have to type in your name and your serial number. Your name should be quite known to you, and the 10-letter serial number can be found on the back side of the disk.
If everything was typed in correctly, another dialogue appears, in which you can choose the components you wish to have installed (at the moment only **complete installation** can be choosen).
By clicking on the button below **destination directory** you can choose the directory on your hard disk to install Smurf to. Then klick on **Start installation**.

If you have the hypertext system ST-Guide installed, you're going to be asked for a destination directory for the hypertext files. Anything should be explained by the dialogues themselves. Just follow the instructions.

If everything is OK, another dialogue should appear, that shows you the current installation status. The upper process bar shows the process for the current file displayed on top of the dialogue, the lower bar is for the complete installation process.

The names of the files and directories should be self-explaining. Importers have the extension .SIM, exporters .SXM, edit modules .SEM, dither modules .SDM and Plug-ins .PLG.

The resource file SMURFICO.RSC in the program directory contains two sets of desktop icons for Smurf and modules.

3 Menu bar and dialogues

"Where is that going to end?"

Olaf after looking at the to-do-list

Like any GEM-Application should do, Smurf provides a GEM-Menu bar.
Almost any entry is accessible by a keyboard shortcut. The standard shortcuts are right behind the menu entries, any shortcuts can be changed with a resource-editor, if you should find one or another inappropriate.

Closing a dialogue window or pressing the **Cancel** button cancels all changes, if they've not been applied by the **Apply** button which applies the changed options without closing the window. **OK** will close dialogue windows as you're used to and apply the changed options.

3.1 Smurf

Like it also should be in any GEM application, the left menu contains, apart of the desk accessories, an entry for opening the program information window ("Smurf it!").
The info dialogue contains various information about the program, who did it, what was used when it all was done, and what version you're using. It didn't get much more interesting.

3.2 File

3.2.1 New image

"Redo from Start"

?

Shortcut CTRL-N
Even without importing data, Smurf is able to view an image. Well, this image will be quite white, because it's brand new and has been generated especially for you.
You can choose width, height and color depth (in bits) for the new image. The size can be in Pixels, mm, inches and DTP-Points (1/72"). Relevant for the last three measurement units is the dpi-resolution. A higher resolution results in more pixels: if an image is, for example, 1 inch in width and height, at a resolution of 300dpi, the image is 300 pixels in width and height. At 600dpi it is 600x600 pixels.

3.2.2 Open

"PRESS PLAY ON TAPE"

C64

Shortcut CTRL-O
This menu entry, guess what, loads an image file.
A file selector will open, in which the image file can be chosen as you're used to.
Afterwards, Smurf will check the file extension and call the matching import module, in case one of them "knows" the file extension. If the extension is unknown to all import modules, Smurf will by request call one import module after another. They'll take a close look at it and decide if they can handle the format or not.
If the image file recognition was successful, the import module will decode the image and Smurf will mostly open a window and display the image on screen.
Mostly because it could happen that either the system can not provide any more windows or there's not enough memory for displaying the image.

For displaying e.g. a 24 bit TIFF image containing 250.000 different colors in a screen mode with 16 colors, the image display has to be reduced (how to choose the type of reduction and what types are provided can be found in chapter 3.3.1).
Depending on the color reduction method and the screen graphic mode, this will result in more or less severe loss of quality.

But Smurf will always keep the original image in memory, so that editing and exporting will keep the original color depth (saving of course only, if the export image file format supports the original color depth). The screen color reduction can, depending on the selected method, take a little while. Dithering a 640x480 image from 16.7 millions of colors to 256 colors with the default "Fast Diffusion" method will take about 9 seconds on a normal ST - which is quite good in comparison to other graphics applications. Users of a Falcon030 will sooner or later benefit of DSP routines - the question is, when we find the time...

3.2.3 Revert

Throws away the active picture and loads it again with the actual preferences. This saves closing and loading with the file selector by hand in two steps. A request if the eventually edited image should really be closed will only take place in non-professional mode. In the professional mode you'v to pay attention to this.

3.2.4 Multiconvert

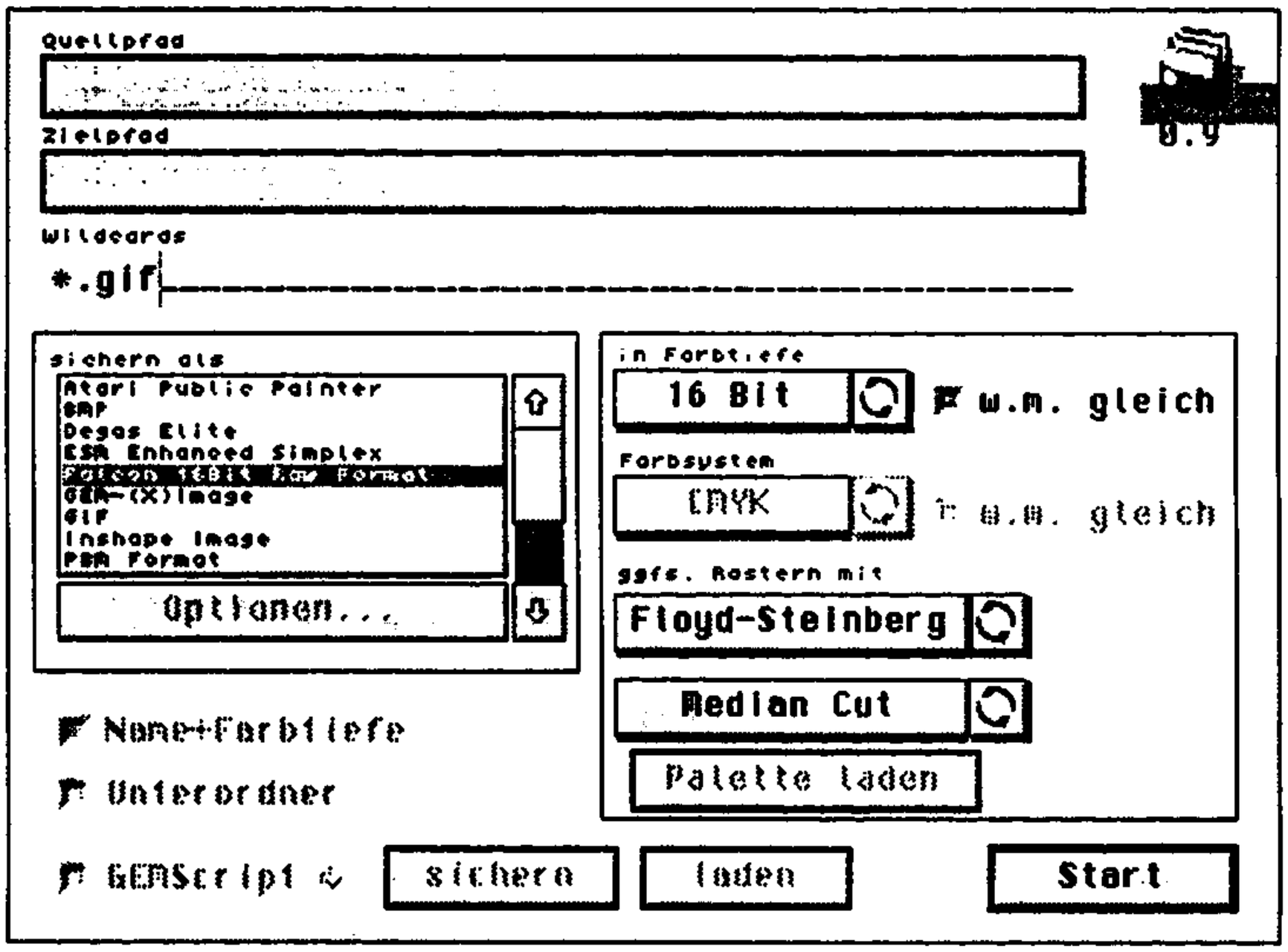
Shortcut SHIFT-CTRL-E

Smurf is an image file converter, and so converting graphics should be one of its main tasks. When it is only about one or two pictures, this can be done easily manually, whereas the conversion of a whole PhotoCD to GIF and PICT could probably be nerve-racking.

Well, and again no problem for Smurf, that's what the Multiconverter is for.

For memory saving purposes the Multiconverter is a non-resident plugin. For more informations about plugins please read chapter 1.4.

This is the dialogue ...



and that's how it works ...

The top button is for the source path, where the images to convert are loaded from. Below it, the destination path, where they're saved afterwards.
You can type up to 64 comma-separated wildcards into the text field. The use of * and ? in any combination is allowed - you can, for example, type in "*.gif,source?.tga" to convert all image files with the extension "gif" and all TGA-images with a filename beginning with "source" and **one** letter or numeral afterwards. To convert all images from the source path, just type in "*". Important is, especially for users of extended file systems, that "*.*" matches only on image file names that have a dot in it.
From the **save as** list field you can choose the export file format. This works as you're used from the export list dialogue (CTRL+M). The **options** button is also the same - it opens the extended configuration dialogue for the selected exporter.

The right lower half of the dialogue can be used like the normal export dialogue which is opened by holding down the ALT key while pressing the startbutton in the export list dialogue (for closer explanations see chapter 3.2.6).
Well, there is a difference: The **keep** switch. That means "keep the source image's color depth, if possible". If turned on, the Multiconverter will try to save the image in the same color depth as it was originally loaded, without color reduction. For not all image

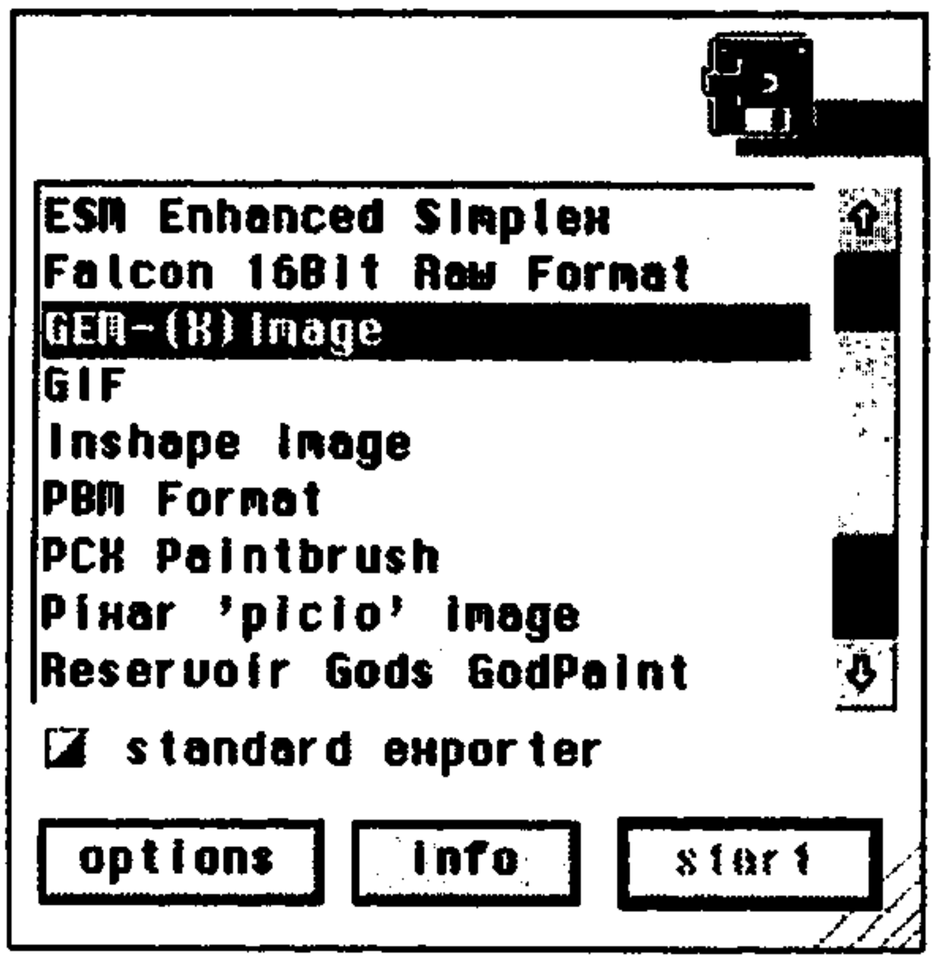
file formats support any color depths (e.g. GIF, which supports a maximum of 8 bits) this can not be done in every case - only if possible.

3.2.5 Close

Shortcut CTRL-U
Well, what to write here? If windows are opened, they have to be closed sooner or later, and this can be done here. This menu entry closes any window, no matter if an image or a dialogue is in it.

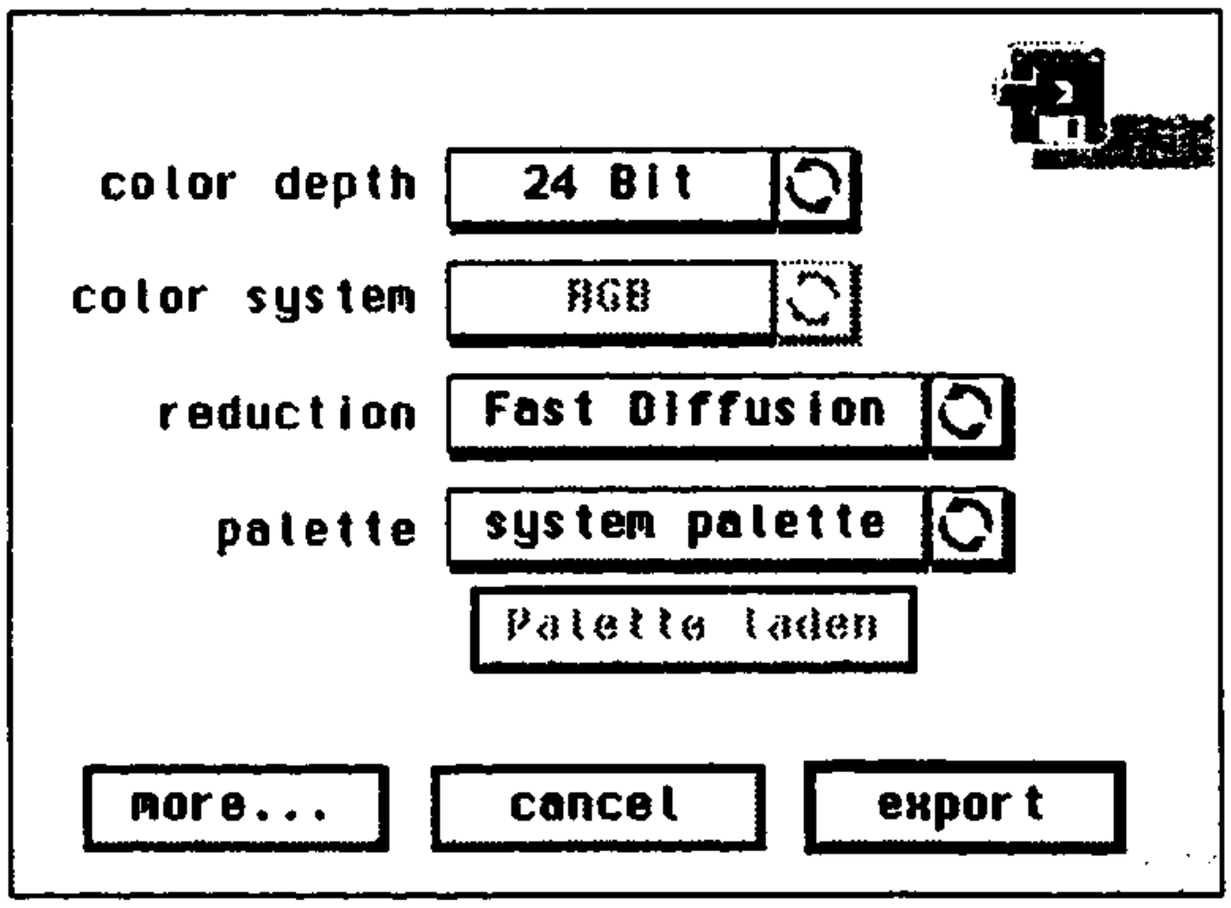
3.2.6 Save as

Shortcut CTRL-M
The accompanying listfield **Smurf Exportmodules** is for managing and starting the export modules. You can find there all active modules from the subdirectory **Export**, that means all modules with the extension .SXM.
Info switches to an info page with informations on modulname, moduleversion, author(s), filename and colordepths.
Double clicking on the desired exporter or pressing **Start** starts the chosen module.



If the selected image file format supports extended specific configuration, such as different compression methods, another window can be opened by pressing the **Options**. The options adjusted in this dialogues will be kept until you quit Smurf or can be saved on disk using the corresponding button.
If the picture shall be converted to an other depth while exporting, there is another formular. It'll be called when holding down ALT while starting a exporter.
This dialogue contains elements you're going to see again in other parts of Smurf too. Like popups for choosing the destination depth and the both popups for choosing the dithering method and the palette mode if necessary. These methods will be explained in depth in chapter 8.

Is it necessary to convert the desired pic to a lower depth than adjusted, or didn't you use this dialogue, the in **Options/Conversion** chosen options will be active. Explanations about this you can find in chapter 3.5.1.4 in this document.
Pressing **more** you get the same extended options as via **Options** directly from the list.



The process of encoding can mostly be watched in the busybox window. At the end a file selector will open, in which you can choose the path and file name to save the image file to.

3.2.7 Print

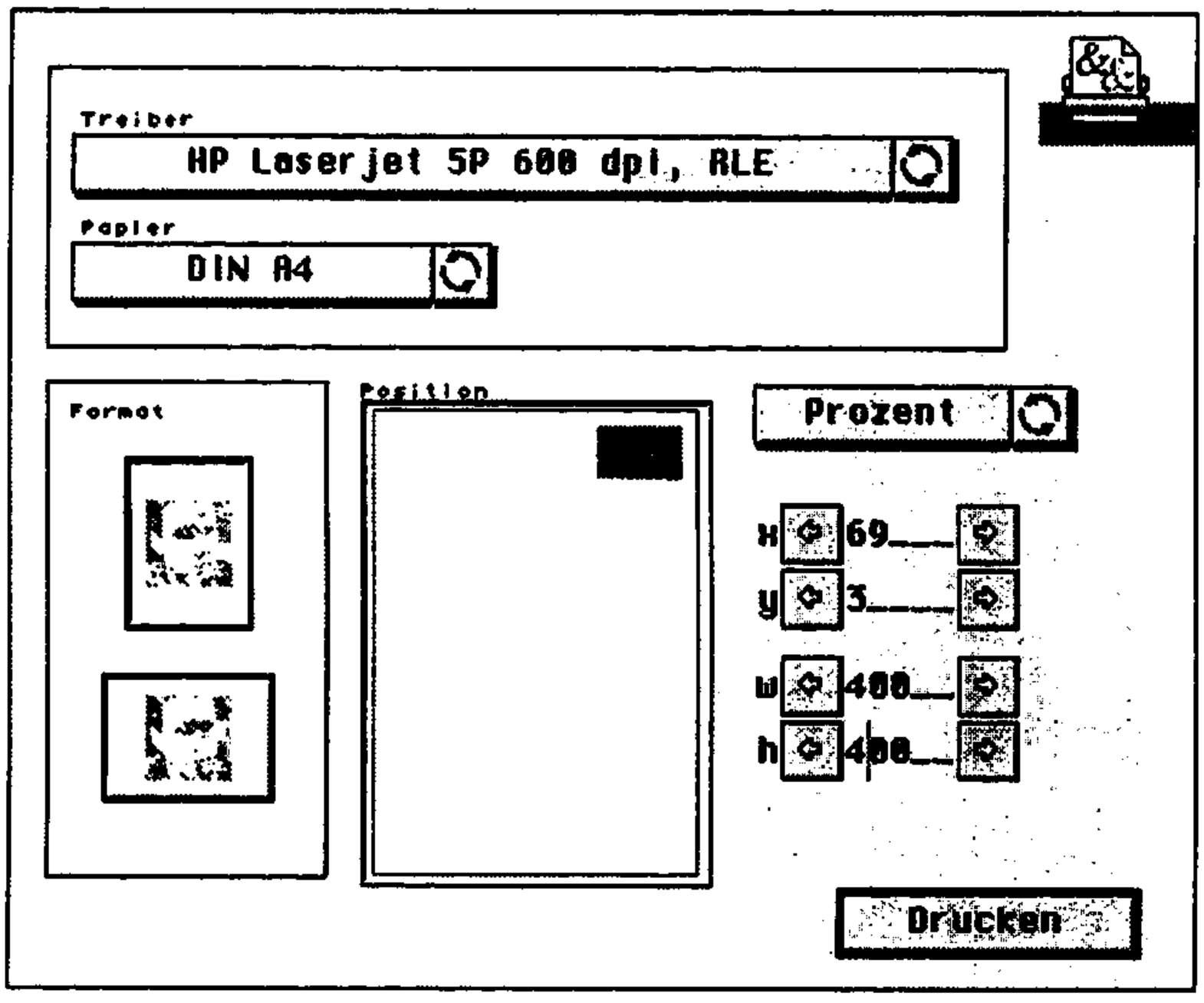
Shortcut CTRL-P
This program function is, like the multiple conversion, a non-resident plugin. For more informations about plugins please read chapter 1.4.

If you like a picture that much, that you don't only want to have it on screen, but also on paper, you can print it out with Smurf. We decided, not to develop our own printer drivers, rather than just controlling the GDOS and leave the work to it.
If you don't have a GDOS, get one. This may sound hard, but it is the cleanest, most compatible and easiest solution for us, and for you, too.

Nevertheless, you have the choice. The choice between our own printing dialogue and the one provided by WDIALOG.
The printing functionality is contained by two different plugins. **PRINT.PLG** provides our own printing dialogue, **PRINTWD.PLG** uses the routines of the system tool WDIALOG. An explanation of the WDIALOG printing dialogues will not take place here, this is the programmer's business. Our printing dialogue should be self-explaining, but we'll do it anyway. The **Driver** popup lets you choose the printer driver. The popup **Paper** contains all paper formats for the selected driver. Default is **default**, which means the standard paper size chosen from the NVDI DRIVERS.CPX.

Previeu shows you, what size and position the printed image will have on the paper. Position and size can either be changed by typing in into the edit fields to the right, or by clicking and dragging inside of the preview box.
If the width or height edit fields are left empty and you reposition the cursor by cursor keys or mouse, the value will be computed by the aspect ratio of the image and filled in. Changing the size of the image is only possible, if the GDOS driver is able of scaling images. NVDI drivers are since version 3.0.

Both printing plugins support printing via GDOS in monochrome and true color/greyscale. Only the 8 color mode is not supported (looks lousy in comparison to the true color mode, so what.).
From version 5 on, NVDI is able of dithering for printing, and these features are going to be supported by Smurf in the future.



3.2.8 Image info

Shortcut CTRL-I
This dialogue shows information about the active image: file name, file size, color depth, a.s.o.

3.2.9 Quit

"I'll be back!"

Arni

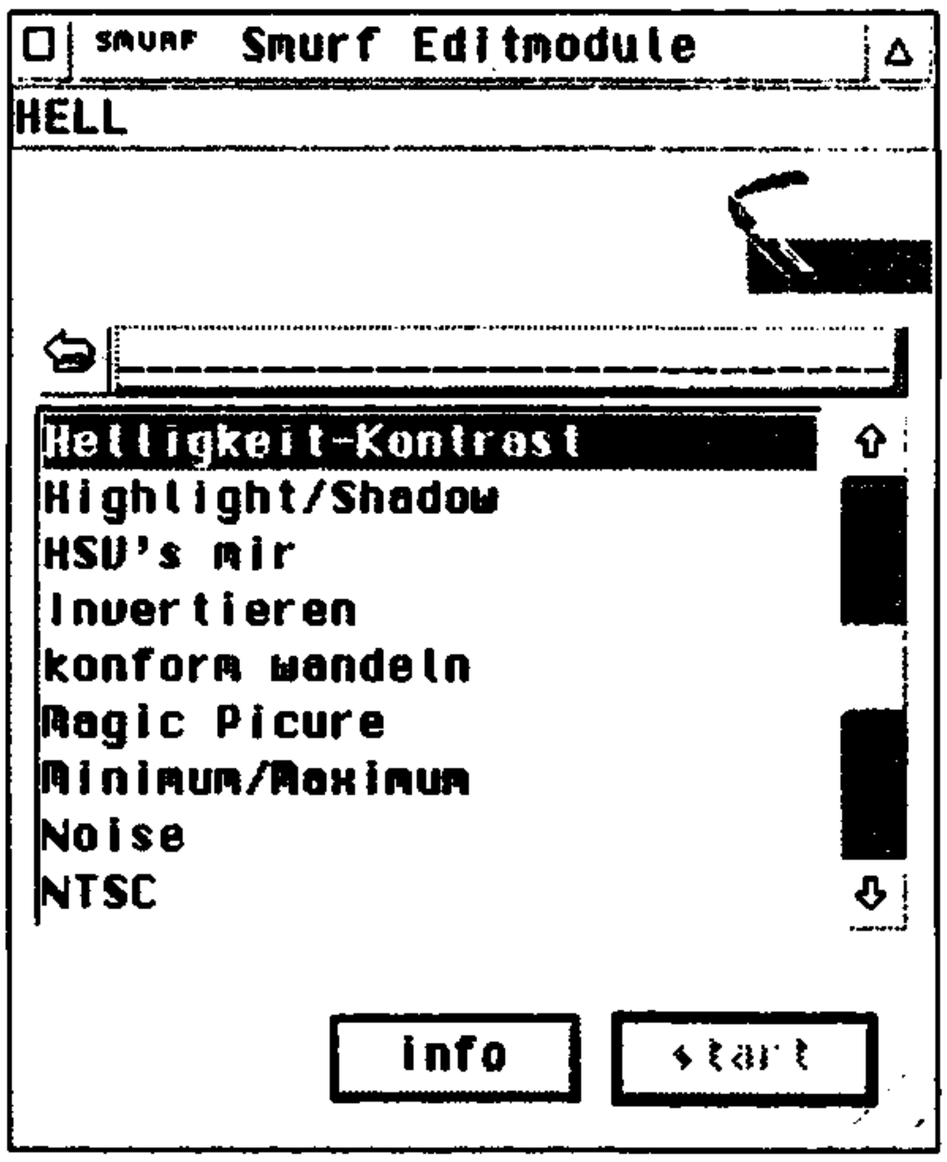
Shortcut CTRL-Q
Smurf closes all windows and terminates.

3.3 Edit

3.3.1 Modules

Shortcut CTRL-B
This menu entry is a big, if not the biggest, difference to a pure image file converter: the image processing modules.
A description of the modules can be found in chapter 6, here we'll only describe the general handling of the Smurf edit modules.

Command centre of the edit modules is the **Smurf Editmodules** list field, containing all active, i.e. with extender .SIM, edit modules that were found in the subdirectory **Edit**.
Info changes to the info page with informations about full module name, module version, author(s), filename and color depth.



Normally a module will always process on the active image. The active image does not have to be the one with the top window, you can recognize it by the selected entry in the pic manager and the light grey toolbar.
If a block selection has been picked up from this image, the block, not the image, is being edited by the module instead!

But: modules can handle up to six different images at a time. If a module needs more than one image at a time (e.g. Bump it up or Spherical Image) the images have to be assigned to the module.
To do this you need the pic manager on screen (and, of course, a number of images loaded). After starting the module, drag the name of an image from the pic manager onto the module window. A popup will pop up, showing you the meaning of the different images the module needs. By clicking on the corresponding button in the popup, you can assign the image to the module.
Repeat this until all images needed (that may depend on the configuration of the module) are assigned. If you want to change an image just drag again and klick on the occupied button. The assigned image will be replaced by the new one.

When starting a module, normally a window will open. Some modules don't open windows (for example "invert", for which a configuration dialogue would be quite unnecessary). To prevent modules like that from being accidently started, a security request will be done. If you're in professional mode (see 3.5.1.1), this request will not take place.

To keep the development of simple modules as simple and the memory use as low as possible, Smurf can provide an internal standard configuration window for the edit modules. This dialogue does not have the elegance of the module specific resources and dialogues, but it has the named advantages. The internal dialogue can only be used by one module at a time, so if a module using this dialogue is running, and another module using it is started, the first module will be terminated and replaced by the second.

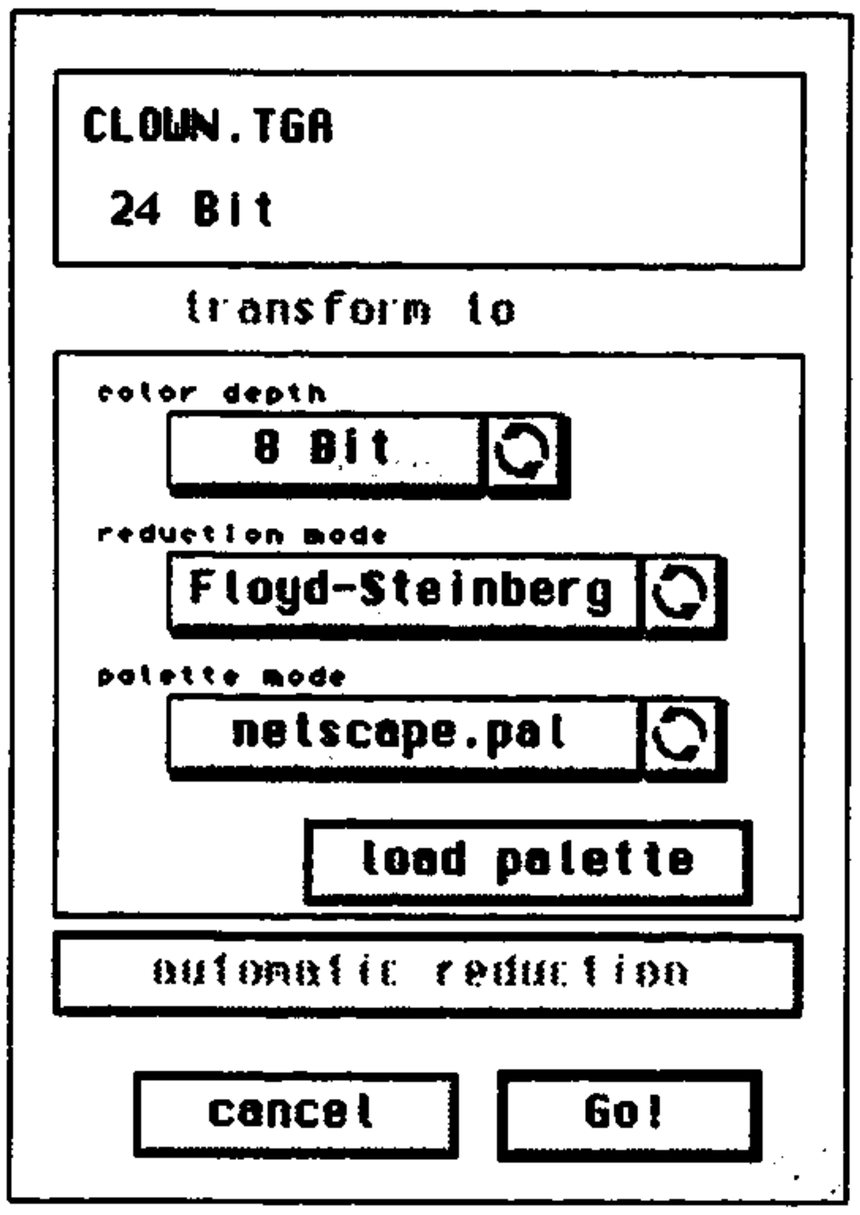
To close a module immediately after it has finished its work, just hold down ALT while starting it.

3.3.2 Transform image

Shortcut CTRL-D

"Another dithering dialogue" I hear you say.

And you're right, another one.



But, where the one explained in chapter 3.2.6 is only for the exported image file, and the one in chapter 3.4.1 only for screen display, this one is for destructive conversion of an image in memory.

For not only on screen you may have less colors than in the image files, but also some image file formats have limitations for their maximum color depth, or maybe you just want to reduce an image in memory to 16 or 256 colors to see, how it looks as a desktop pattern.

This dialogue also contains the familiar popups for palette mode and reduction method. And again here I want to refer to chapter 8, where you can find a detailed explanation of the specific algorithms.

3.3.3 Duplicate image

"The same again."

The man in the bar, who doesn't want to let the barkeeper go home.

Why would you need the same image two times? No idea - but there has to be a very good reason, why so many people wanted this feature.
After selecting this menu entry, the active picture is copied and displayed in a second window. A block eventually picked up in the image is not duplicated!

3.3.4 Block operations

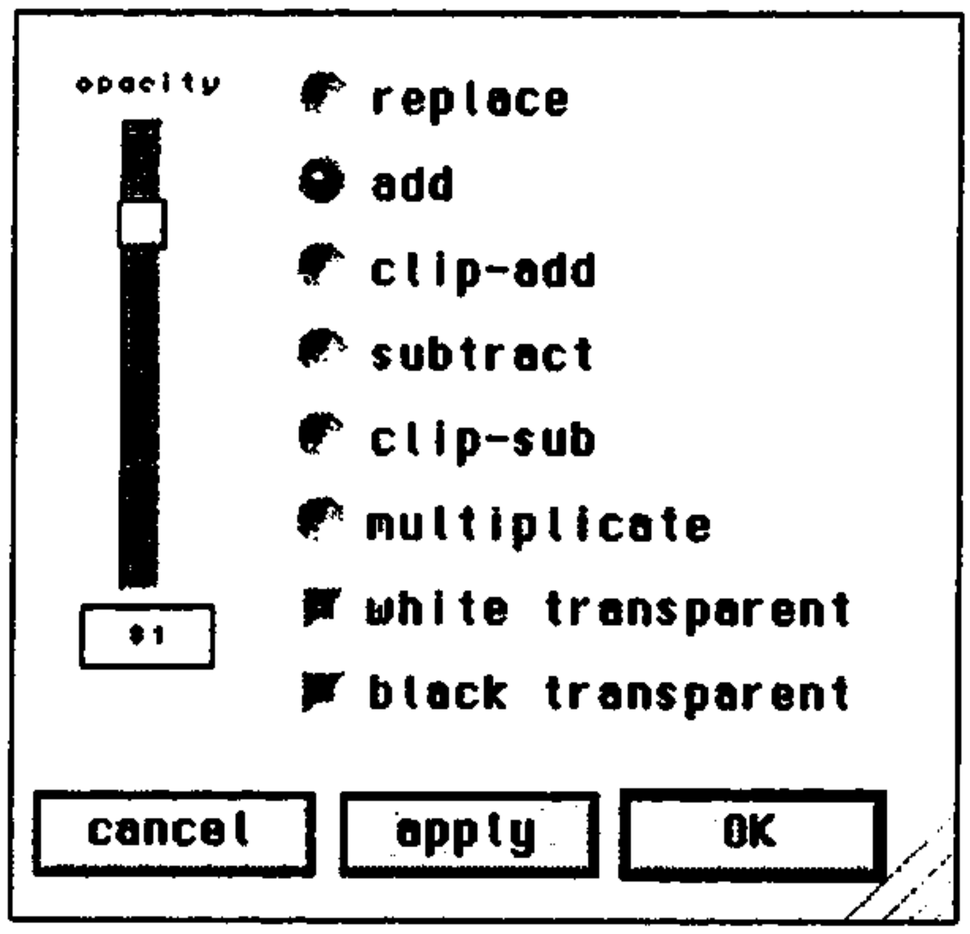
By left clicking into a picture window and holding down the mouse button, you can open a block frame. The size of the block frame can in the true sense of the meaning be changed by pulling on all edges. The current mode is visualised by the mouse pointer:

- ↔ means, the frame size can be changed in horizontal direction at the current mouse position.
- ↕ means the same in vertical direction.
- ✥ means, the frame size can be changed into both directions (appears, when moving the mouse pointer over a corner of the frame).
- ✋ means, the whole block frame can be moved into all directions.

It has to be remarked, that the frame can be positioned partially outside of the image. But all block operations (copy, cut, insert or pick up) will only operate on the areas of the block frame, that intersect with the image rectangle.

The copy, cut and paste functions only refer to the active image, respectively the block of the active image. If a block is picked up into the frame, all image manipulations operate on the picked block area, not on the source area in the image itself.
The **copy**, **cut** and **paste** operations work with both, empty block frames and picked block selections. For transmitting an image block to another image or to other applications (via Drag&Drop or AV-protocol)or for editing a block area, the block has to be picked up by double clicking into an empty block frame. Double clicking into a block frame with a picked area will result in insertion of the block into the image at the current block frame position.

For this insertion there are the different modes **replace**, **add**, **clip-add**, **subtract**, **clip-sub** and **multiply**. All modes work with free adjustable opacity and the switches **white transparent** and **black transparent**. The configuration dialogue can be opened by selecting "mode" from the block popup menu (see there for further instructions).



The **replace** mode will just replace the image pixels by the block pixels. By turning down the opacity the block will get more and more transparent.
The **add**, **subtract** and **multiply** modes just do what they are with the source- and destination pixels. But, subtracting one pixel value from another or adding them together can result in negative or overflowing values (>255). Overflowing values are "jumping" from 255 to 0. That means that 293 + 37 is not 276 but 21 - this can have quite silly looking results, but this even may be on purpose sometimes. If it is not, the values can be clipped to a range between 0 and 255 using the modes **clip-add** and **clip-sub**. Then 239+37 is still not 276, because one byte just ranges from 0 to 255, but the highest possible value represented by a byte: 255. Same for results that actually would be negative, they are clipped to the lowest possible value represented by a byte: 0.
All this sounds quite confusing - just try it out.

The switches **white transparent** and **black transparent** can be of great use, if you, for example, want to insert text generated with the text module into an image without the white background of the text. These switches just make the colors white respectively black being ignored during the insertion.

3.3.4.1 Select all

Shortcut CTRL+A
Selection of this menu entry will open a block frame in the active image exactly as big as the image rectangle, positioned on the coordinates 0/0. A block frame eventually already inside of the image will be deleted, no matter if it contains a picked area or not!

3.3.4.2 Copy

Shortcut CTRL-C
The selected area of the image or a picked image area is copied into the GEM clipboard.

3.3.4.3 Cut

Shortcut CTRL-X
The selected area of the image or a picked image area is copied into the GEM clipboard. Additionally, the selected area will be deleted from the image, resp. a picked up image area will be removed.

3.3.4.4 Paste

Shortcut CTRL-V
Inserts the clipboarded image into a new block frame of the active picture. A block frame eventually already inside of the image will be deleted, no matter if it contains a picked area or not!

3.3.4.5 Crop

Shortcut CTRL-R
Removes anything that is not selected from the active picture. Only the selected area is kept.

3.3.4.6 Remove block

Shortcut CTRL-H
Removes the block frame and the eventually picked image area from the active picture.

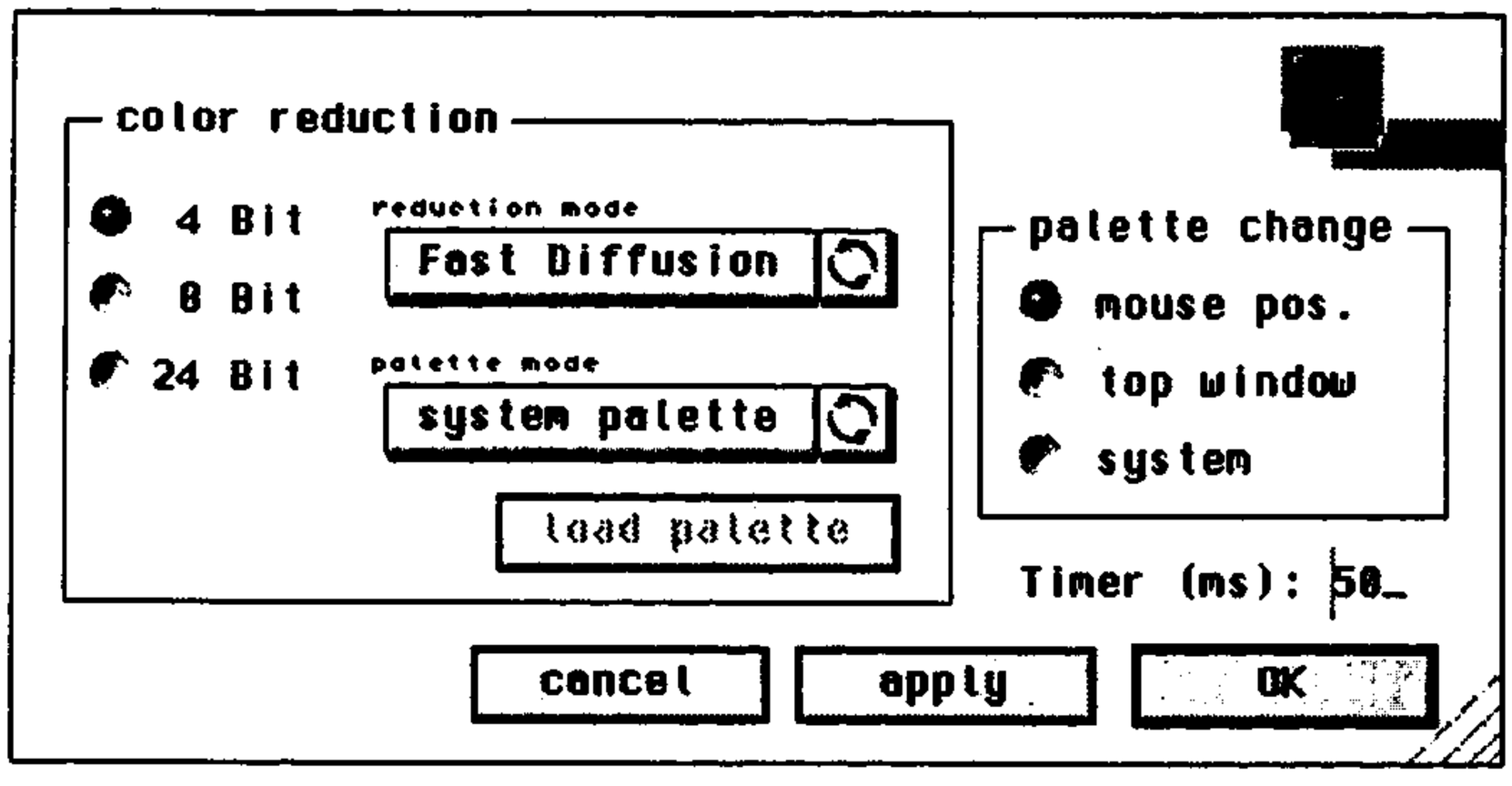
3.4 Display

3.4.1 Configuration

"On screen! Maximum enlargement!"

Captain Picard

This dialogue configures the quality and speed of the display of images on screen.



Instead of configuring all image types at a time, we decided to implement a configuration of three image color depth ranges.
These ranges are: 2-4 bit, 5-8 bit and 16-24 bit. Monochrome images can not be configured, because there is no need for dithering them.

The three ranges relate to the **picture** depth, not the **screen** depth!
The three radio buttons on the left side toggle between the three color ranges. The separation of the configuration in the different renges has the reason that dithering a 24 bit image to a 16 color screen mode is best with a Floyd-Steinberg algorithm. For a 16-color image a nearest color conversion is mostly good enough.

The popup menus to the right of the radio buttons configure the dithering method and palette mode for the selected color range.
That after opening the dialogue 4 bit is active although you left it with 24 bit has no deeper meaning, it's only opened the way that 4 bit is visible.

The palette modes implemented in Smurf are:

1. system palette
2. median cut
3. fixed palette (from file)

and these are the available dithering methods:

1. Fast Diffusion
2. Floyd-Steinberg
3. Ordered Dither
4. Nearest Color
5. Greyscale

Will Floyd marry Steinberg? Why is Nearest Color so ugly? What does Ordered Dither do at night in the basement? And who the hell is John Doe?
See chapter 8, "Mr. Dither on the run", explaining something of the above.
Knowing the differences is necessary for general understanding and the right choice of the appropriate methods, but not necessary for explaining the handling of Smurf and hence discharged to chapter 8.

Dithering to system palette does definitely not have the results with the best quality, because dithering will be done with the system colors from the control panel. But there are advantages: You can, especially in multitasking environments, recognise anything on screen at any time, because the system colors will not be changed; apart of that, all images loaded and dithered to system palette will be recognizable at a time, because the same palette is used for displaying all of them.
Furthermore the system palette is well known to Smurf, and because of the NCT file generated on startup the dithering to system palette is a lot faster than e.g. to a median cut palette.

But to prevent an image color palette different from the system colors from muddling the whole system permanently, the following three methods for switching between image and system palette are offered:

1. Mouse position.
The active palette is going to be set in dependence of the mouse pointer position. If the mouse pointer is moved onto an image window, the displayed system palette is going to be replaced by the image palette. Moving the mouse pointer away from the image window will switch back to the original system palette.

2. Top Window
The system palette is going to be replaced by the palette of the image in the top window. If a dialogue window or another application is topped, the palette switches back to system colors.
3. System
The system colors will always be active, no matter where the mouse pointer is or what window is topped. This can result in incorrect display of images dithered to median cut or file palette.

The methods 1 and 2 have in common, that they only work, if the top window is one of Smurf's windows, so if Smurf is the active system process. It's better like this, believe us.

OK, how do these configurations work?
First of all, the latest configuration made will be used on all images opened afterwards. After editing or transforming an image, the color depth dependent configuration will be used. ReDither also uses the latest set options - except, if you hold down SHIFT while clicking on ReDither. In that case, the screen display configuration dialogue will open, with the color depth range matching the image you selected ReDither from.

The timer sets the reaction speed of the display. It can be useful to increase the timer value on slower computers and get an update of the coordinates display, color palette, a.s.o., only every 100 or 150 ms.

3.4.2 Cycle image/dialogue

Shortcut CTRL-W / SHIFT-CTRL-W
These menu entries cycle through the windows. But who wants to cycle through all dialogues, when searching a certain image window or cycle also through the image windows, when only wanting a certain dialogue window to be on top?
See? That's why we made both window types cycleable separately.

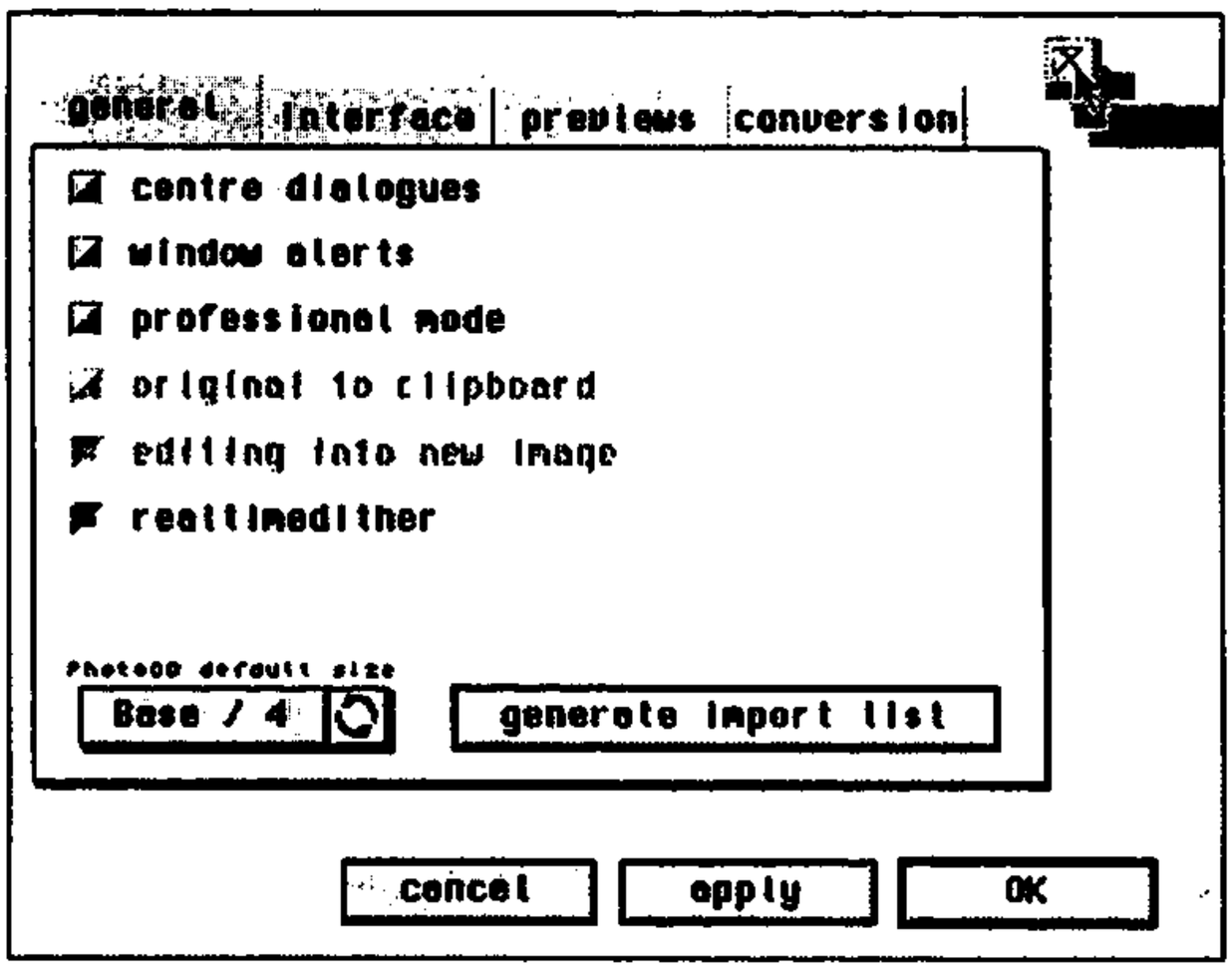
3.5 Options

3.5.1 Options

Shortcut ALT-O
This dialogue allows the configuration of the look&feel of Smurf. For lucidity purposes we decided to split the dialogue into four registers.

Independent of the selected register, **Generate import list** will generate an import list (who'd believe it, sometimes our notations even are the root of the matter). Normally, the import list will be generated and saved at the first start of Smurf. But if you get new import modules or remove old ones, the list has to be generated again by pressing this button.

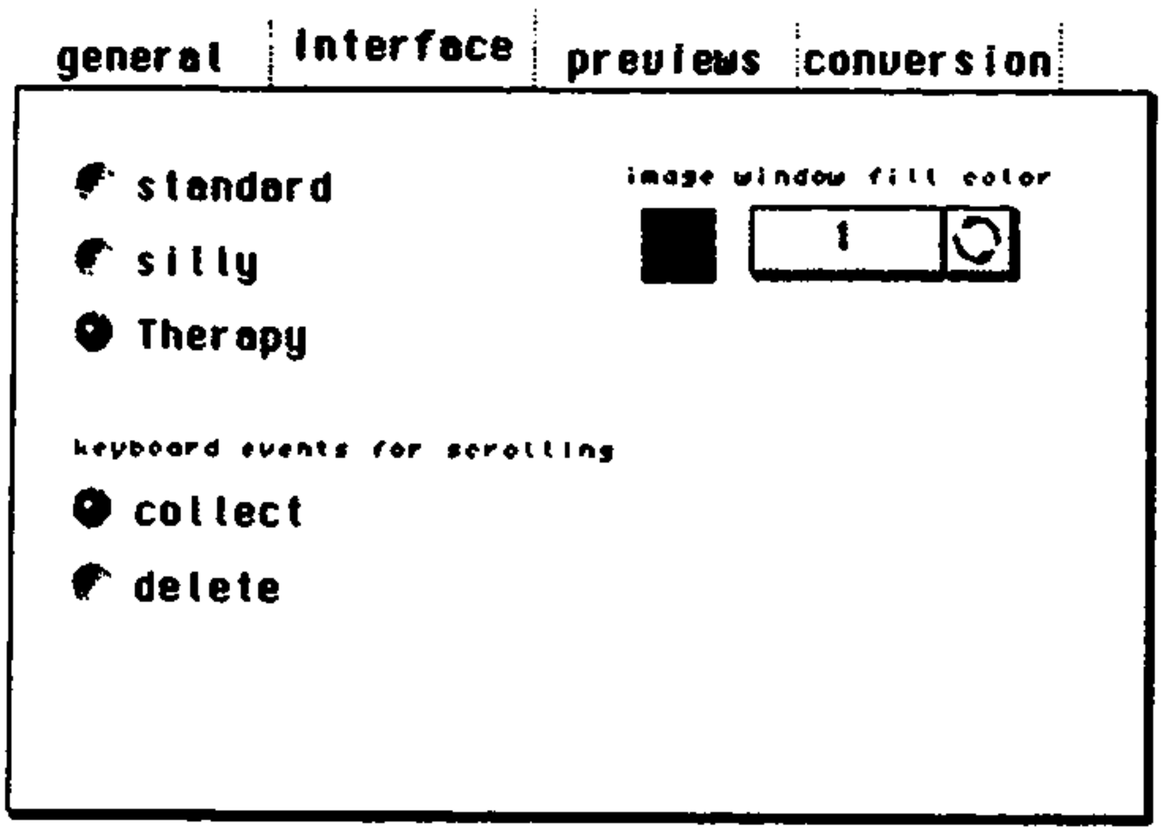
3.5.1.1 General



Centre dialogues: This option centres all dialogue windows, if the position has not yet been saved. Otherwise the dialogues will appear in the upper left corner of the screen.
Window alerts: This option will force almost all alert boxes into a window. Window alerts are application modal, meaning, while open you can work in other applications but not in Smurf.

Professional mode: This mode will turn off all warnings and messages concerning automatic image conversion. That means, if e.g. an edit module will force an 8 bit image to be converted to 24 bit, no request will appear when this is turned on. Only errors or important questions stay turned on.
PhotoCD: This popup selects the size for loading Kodak PhotoCD images (PCD image files contain the image in more than one resolution). But, if you just wish to load one or another PCD image with another size, you do not always have to reconfigure that. Holding down the ALT key while loading a PCD image will open this popup, too, allowing you to select the size for loading the current PCD image.

3.5.1.2 Interface

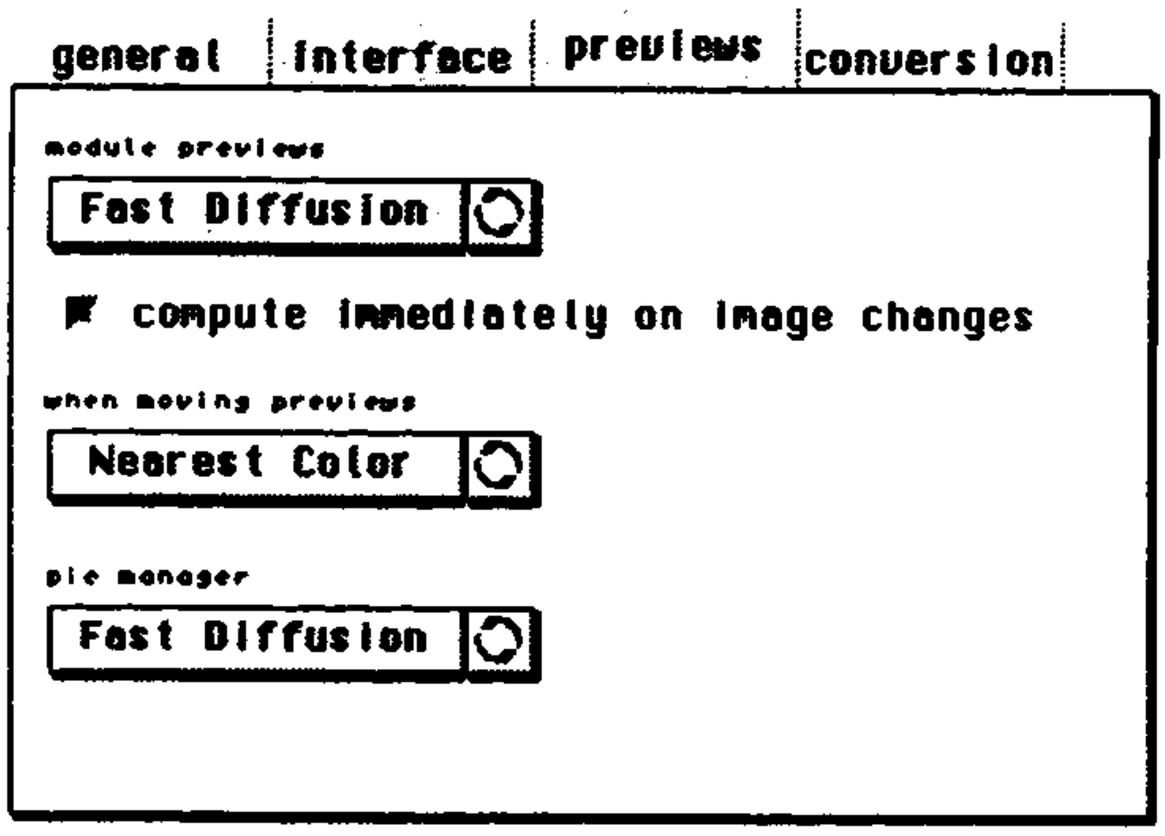


Interface - Standard/Silly/Therapy: Selects the look of the user defined objects: radio buttons, checkboxes and popup cycle buttons. The three configurations have to be understood as an increase of silliness.

Image window fill color: This popup selects the color, that will be used to fill image windows outside of the actual image rectangle.

Keyboard events when scrolling selects, what happens with multiple cursor key presses in image windows, that cannot be handled immediately. **Collect** means, the keypress events will be collected until there is enough time to process them, and then scroll all at a time (results in an increase of the scrolling speed the longer the cursor key is held down). **Delete** means, there will always be only one keypress operated, scrolled and then the key buffer is deleted. Both methods prevent image windows from scrolling after releasing the cursor keys.

3.5.1.3 Previews



Module previews: With this popup you can choose the method for dithering the previews in module dialogues independent of the screen dither. Only algorithms capable of system palette dithering can be selected here.

in movement: This popup selects the dithering method for module previews, while you are moving them with the mouse. That means, you should choose a fairly simple and quick method, that allows you to move the previews fluently - we recommend nearest color. Only algorithms capable of system palette dithering can be selected here.

pic manager: This popup selects the dithering method for the thumbnails in the pic manager window. Only algorithms capable ... you know the rest.

compute previews automatically: If a configuration value is changed in the internal module dialogue, the preview is going to be updated automatically, without the

need of clicking on it. The disadvantage is, that, especially on slower computers, the computation of the preview may take a little while and you will have to wait for it.

3.5.2 Save configuration

Shortcut ALT-S

Saves the current configuration, like dithering methods for screen display, positions of opened dialogues and all options adjusted in the options dialogue (3.5.1) to the file SMURF.CNF in the directory specified by the environment variable HOME. If HOME is not specified, the Smurf program directory will be used, instead.

4 Other dialogues

"Everything except for this one plus the few little other one, there."
unrecognised coder after uncounted sleepless nights.

4.1 Busy box

Shortcut ALT-F

We called this little window with the horizontal bar busybox, 'cause it tells, when Smurf is busy, and how long. Well, just open it now.
The bar is called the busy bar and will grow from the left to the right when Smurf or one of his modules is working. So you're always going to know, how long it will take until the work (filter operation, image conversion, a.s.o.) is done. An exception is the computation of previews. Because they're quite small, it will normally take very short time to calculate them, and a process display is unnecessary then, so the busybox is switched off during preview computation. When a module needs more than one second for computing a preview, it will be turned on again.
The look of the busy bar can be toggled by clicking on it. The actual look will be saved, when saving the configuration (ALT+S).
Above it, the status text, which is "OK" when Smurf is ready and waiting for the next task. When importing, exporting, dithering or filtering an image, it shows, which module is working at the moment, and/or what it is doing. And, if you're pretty quick-eyed, you can maybe see, what it says when you quit Smurf.
Below the busy bar, you always see the free Memory of your computer (total of ST- and FastRAM).

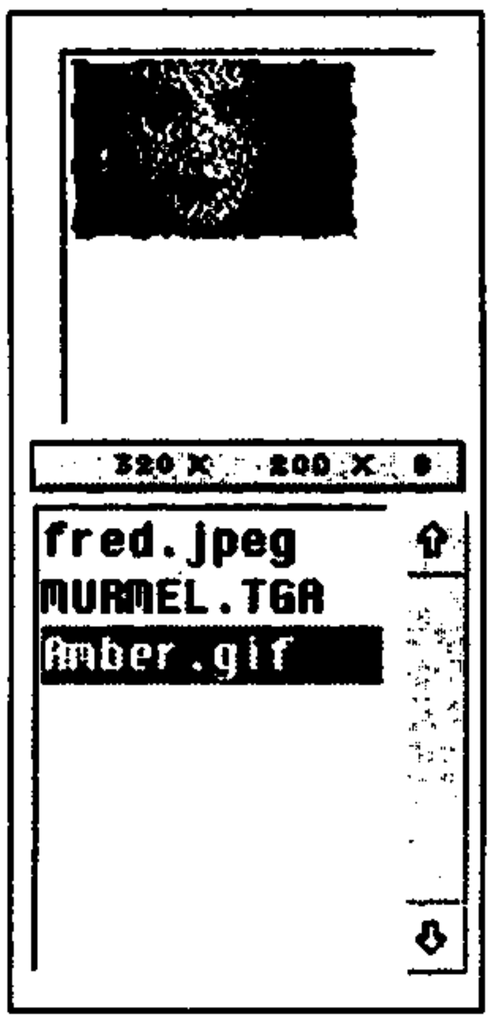
4.2 The pic manager

Shortcut ALT-M
We're happy.
We're not happy for you, rather than for ourselves, to have our own manager.

The pic manager is little, slim and important. So much important, little and slim, that you should always keep him on your monitor. This slim, little and important dialogue not only helps you to keep control over all images loaded, it is even indispensable for some operations.

4.2.1 What the pic manager can do

All loaded images are listed in the Pic manager window.
If you've lost trace of an image window (maybe you've opened too much of them or you've iconified some of them), a double klick on the entry in the manager will top it.



And then, there is the box on top of the window - the thumbnail box. You forgot, what the picture called **AMBER.GIF** or **MURMEL.TGA** looks like? Just klick on the entry and Smurf will hurry to generate a small thumbnail display of the image.
Also for users, that just enumerate images in various states of work, the thumbnails are a good way to keep an overview.
The size of the selected image in pixels is always displayed in the text field below the thumbnail box. Even less disorientation, quite good, eh?
In spite of the fact, that in the pic manager, the whole image is shown as a scaled, tiny preview, the image will not always be visible in its full size inside of the image window. To keep orientation, a box in the thumbnail will always show you the visible rectangle of the image. By clicking into the box and dragging it, you are able to navigate even in big images.

4.2.2 What else the pic manager can do

You don't like the order of the loaded images (e.g. when cycling through the image windows or when generating a movie loops, this will be implemented in later Smurf versions...])? Just take an entry, drag it and drop it, where you want it to be.
Furthermore, the pic manager is not only important, but indispensable for the use of modules, which are able of handling more than one image at a time (e.g. Bump it up or the Channel mixer). The images used by the module, have to be assigned from the pic manager (see chapter 3.3.1 for a detailed description).
Also for Drag&Drop operations with other applications, the pic manager is of good use. Two further features can be used in combination with ALT and CTRL. CTRL-double klick sends the image to the AV-server, mostly the desktop, and asks him to view the image with a viewer, which is assigned to the file type. ALT-double klick will ask the AV-server to open a directory window with the image file.

5 Image windows

"Life is just a lousy adventure game.
... BUT THE GRAPHICS IS GORGEOUS!!!"

Each time, an image file is loaded, Smurf will open a window, displaying the image.
For the display, the image will mostly have to be dithered to the screen color depth, which will be done by the configuration from the **Display/Configuration** dialogue. We recommend the "Fast Diffusion" Algorithm, a simple error diffusion method. This is a dithering with high speed and relatively good quality.

5.1 The tool bar

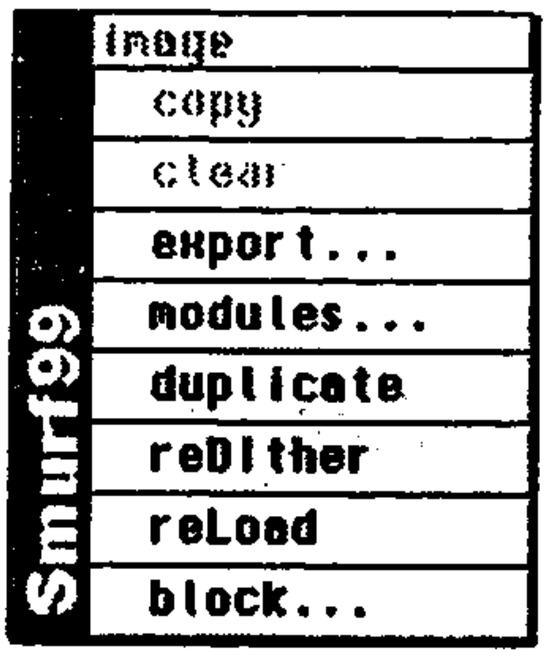
On top of every image window is this tool bar:

It continuously displays information and provides two functions:
The question mark to the left will open the information window for the image. Right beside it is the zoom button, which will open a popup, in which you can select the screen display size.
If you think, an image you want to load, is very big and not only takes a long time for being displayed, but also covers much of the screen, you can choose the display scaling for this image when loading it. This is done by holding down the CONTROL key when loading the image. Before the screen display is being generated, the zoom popup will open, and you can select the display scaling.
But: the scaling is not being applied to the image itself. Only the screen display is scaled!
Further information displayed in the toolbar are the X- and Y-position of the mouse pointer in the image and the size of an eventually opened block frame. These are always the real measurements, independent of image display scaling. In a 320x200 image with a display scaling of 50%, a mouse pointer in the right lower corner will be displayed as 320/200 coordinates.
The three color displays show the R, G and B color values of the image pixel at the current mouse pointer position. In palette images, the color palette index of the pixel will additionally displayed in the "I"-Box.

5.2 the popups

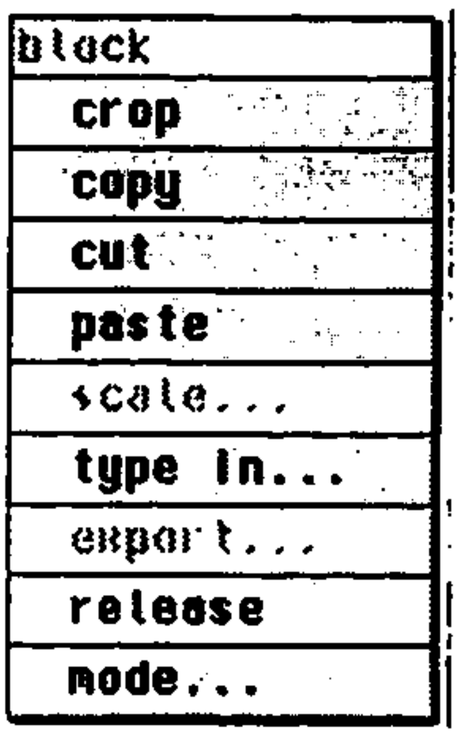
"Wop"

A hundred thousand popups. Or something like that.



A right klick into an image window will open this popup menu. **export...**, **Modules...** and **duplicate** do the same as the menu entries **Save as...**, **Modules...** and **duplicate image**. See also 3.2.5 and 3.4.1 resp. 3.4.3.
ReDither causes a redither of the image with the actual valid settings. Clicking with SHIFT will open the display configuration dialogue, initialised with the matching color depth range.
reLoad causes reload of the image. See paragraph 3.2.3 for further information.
block... opens the popup menu shown below.

Right klick into a block frame or selecting **Block...** in the image popup will open the block popup.



crop, **copy**, **cut** and **paste** will do the same as the corresponding menu entries, see 3.4.4. Also see there for block operations.
type in... a dialogue opens, in which size and position of the block frame can be typed in precisely.
release removes the block frame, including an eventually picked image area.
mode... opens a dialogue for configuring the block write modes and opacity for inserting a block into an image.

Which popup will open by the right klick, can be inverted with the SHIFT key. Clicking right into a block frame with the SHIFT key held down will open the picture popup, outside of a block frame, the block popup.

6 The edit modules

"The brother's gonna work it out."

The Chemical Brothers

The edit module port makes Smurf more than just an image file converter. Together with the 46 edit modules, it provides quite powerful possibilities for image manipulation.

Many of the modules can not handle all possible color depths, but expect to get the images in a certain, normally high, color depth. This is not because of the laziness of the developers (well, mostly not ;-)), but has technical reasons. Filters, interpolations, fading effects or just inserting a block into an image can result in new colors, that are not in the original image. In images with only 16 or 256 colors, there is a high probability for new colors being generated when operating with a filter or inserting a block. And that's why images will mostly be expanded to 24 bit color depth. The 16.7 millions of colors will be enough for any operation.

1. Mosaic
2. Frame
3. Bump It Up
4. Chanel No. 5
5. Channelmixer
6. Clip'n'Pic
7. Displacement
8. Drehen 90°
9. Drop Shadow
10. Edge-O-Kill
11. Equalizer
12. Expander
13. Color gradient
14. free 5x5 Filter
15. Gamma correction
16. Greyscale
17. Brightness/Contrast
18. Highlight/Shadow
19. HSV it!
20. Invert
21. conform transform
22. Magic Picture
23. Minimum/Maximum
24. Noise
25. NTSC
26. Painting
27. Scatman's World
28. Shear
29. Sinus waves
30. Scale
31. Solarize
32. Speed-O-Move
33. Spherical Image
34. Mirror
35. Twirl
36. s/w threshold
37. Text
38. Smooth
39. Weight of Color
40. Wind
41. Zoom

If a module needs more than one image at a time, the images will have to be assigned to the module by dragging them from the pic manager to the module window. See also chapter 3.3.1.

But now for the description of the modules. A short explanation of the module abilities and the options can be given by the bubble help in the program.

6.1 Mosaic

The effect generated by this module can be done also by non-extrapolating scaling to a smaller size and additional non-interpolating resizing back to the original size.
But, the actions this sentence describes, would be just as long and complicated, as the sentence is. So, save time and work and use this module.
The edit fields adjust the amount of pixels used for the mosaic effect.
Supported color depths: 8, 16 and 24 bit

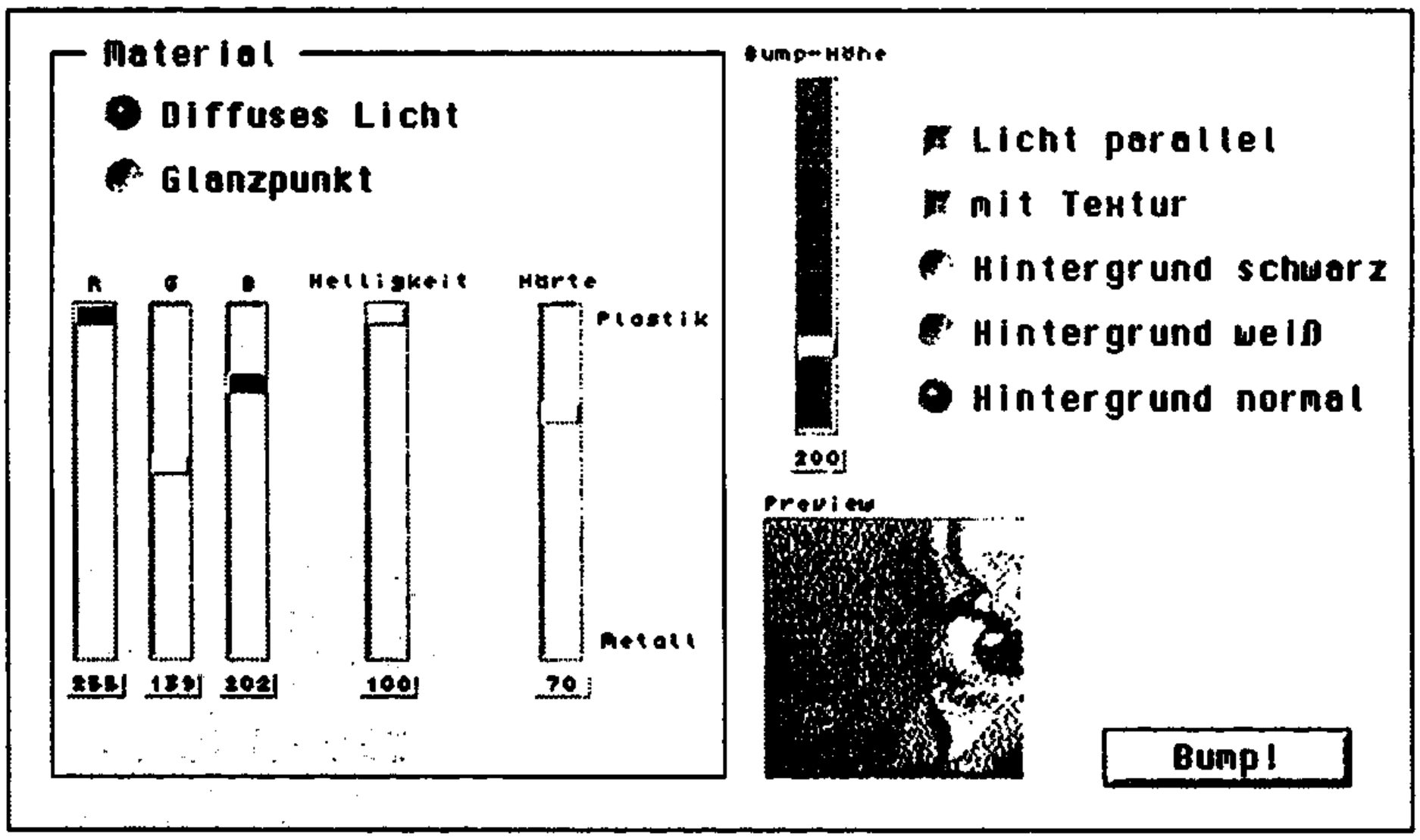
6.2 Frame

You wouldn't hang up your oil paintings without a frame. And to keep your images in your computer from being "naked", this module was created.
It generates a uni-colored frame with adjustable size and color.
Default for the frame color is the one set in the three sliders. But, if you want the frame color exactly to be the color of the top left pixel of the image, you can make it being used by selecting the **use top left** switch.
For positioning the image inside of the frame, there are four text edit fields. They adjust the size of the frame to the top, left, right, and bottom in pixels.
If you want the image to be centred in the frame, automatically, just select the **centre** checkbox. In this case, only the values typed into the **left** and **top** edit fields, will be used.
Supported color depths: 1-8, 16 and 24 bit

6.3 Bump It Up

This module is for rendering lightened surfaces by bump mapping. The bump mapping procedure is mainly used in 3D-raytracing software and generates relief images.
By drag and drop the image "bumpmap" has to be assigned to the module - the bump map image is automatically being converted into greyscale. In this greyscales, bright areas are rendered as "high" areas in the resulting image. The light reflection from the light source (which can be positioned by the crosshair which appears when the images

have been assigned and the module window is topped) is then computed onto the bump map.



When the **diffuse light** radio button is selected, the five sliders in the left half of the dialogue adjust diffuse light parameters, and specular lighting parameters, if **specular** is selected. The R, G and B sliders refer to the color of the light source and/or material, the **Brightness** adjusts the brightness of the diffuse/specular lighting (who'd have thought it ...) and the **Power** is equivalent to the "hardness" of the material - the more, the smaller and more powerful the lighting gradient gets.
The **Parallel** switch turns the point light source into a parallel one, which will turn off the light gradient on the surface, so that the generated surfaces can be used for tiling (e.g. desktop patterns).
The button **Texture** turns on the rendering of a texture image onto the surface. The texture has to be assigned from the pic manager, as the bump map, and is "wrapped" onto the surface. If the texture is smaller than the bump map, it will be repeated as many times as possible to match the bump map size. So in this case, it will mostly be better to use a tileable texture (see also 6.9, "Edge-O-Kill").

The **Background** mode selects the background of the rendered surface, meaning all areas, that are black (0/0/0) in the bump map. In the rendered image, these areas will be set to the selected color (black, white) or included in the rendering (normal).
The **invert** switch inverts the bump map before rendering it. That means, black is high instead of white (the inversion is also used for the background mode.).
This makes the rendering of bumped text quite easy: just write your text with the text module, smooth it, and use the Bump it up with **invert** and **background White** turned on. Supported color depths: 8 bit grey bump map, 24 bit RGB texture and destination image.

6.4 Chanel No. 5

Curiously, this module has got nothing to do with the fifth channel. Accentuated in french, the title could sound quite common, but it has also nothing to do with the perfume. The name was chosen just for fun.
But it has something to do with three channels, the color channels, meaning R, G and B. By a radio button, you can select one out of five modes, which will swap different color channels.
Supported color depths: 1-8, 16 and 24 bit

6.5 Channel mixer

You have to assign two images. Now, you can choose one of the color channels on both sides of the dialogue, each referring to one of the images, the left for the source, the right for the destination. The selected color channel of the source image will then be inserted to the selected channel of the destination image.
Actually done for images that come as three channel files, it is quite good for playing around. Oh, yes, and if you have stereo images, one for each eye, you can make 3D-images for red-green - glasses.
Supported color depths: 1-8 and 24 bit

6.6 Clip'n'Pic

This module is kinda like the opposite of the Frame module. A not very intelligent but functional algorithm "traces" the frame eventually around an image and cuts it off automatically. The frame has to be of only one color, to make it work properly - as said above, the algorithm is not very intelligent.
Supported color depths: 1-8, 16 and 24 bit

6.7 Displacement

This module renders free distortions on any image. The distortion is done by a displacement map, the grey value of each pixel from the map is a displacement value for the destination image's corresponding pixel.
A displacement map pixel's grey value of below 128 will move the destination image's pixel up and right, above 128 down and to the left. For a better predictability of the result you should convert the displacement map to greyscales before starting the module. Photos, e.g. of people or landscapes, are not very good for the use as a displacement map. The best are images of geometrical shape, textures or mathematically computed images, quite good are blurred concentric circles for example.

We recommend the use of tileable images, which can be generated with the Edge-O-Kill module, especially if the displacement map is smaller than the destination image. Sometimes it may be necessary to blur the displacement map with the Soften module before applying it. Using this module means trying it out. For some examples can be found in the gallery of our web site.
Supported color depths: 24 bit

6.8 Spin 90°

The image is rotated by 90 degrees to the left or to the right, or by 180 degrees (here, the question is, in which direction ;-)).
Rotating around 180 degrees and 2-axis mirroring have the same result. The difference is, how both modules do it. The advantage of the mirroring is, that only a buffer memory of the length of one image line is needed, but the whole operation takes more time, whereas rotating is quite fast but needs a buffer in size of the whole image.
Supported color depths: 8, 16 and 24 bit pixelpacked only - GEM standard format images are converted.

6.9 Drop Shadow

This module is floating - just as your images will be.
Drop Shadow generates the well known soft shadows. You need an image with a preferably white background and preferably a foreground of another color. The shadow will be dropped onto the white areas in the image.
The **strength** slider adjusts the darkness of the shadow, **soft** how soft, and **size**, how big it is going to be.
The text edit fields contain the distance of the shadow to the image foreground in pixels. Negative values are also possible.

Supported color depths: 24 bit

6.10 Edge-O-Kill

Did you ever try to tile an image onto your desktop without getting visible edges, where one tile touches the other?
Edge-O-Kill will make these edges disappear and makes smooth tiling possible. It's no kind of magic, it's simple fading. The image borders in opposite are faded into one another. How much of the image is to be faded can be adjusted with the **Pixel X** and **Pixel Y** text edit fields in Pixels, or with the **X-Fade %** and **Y-Fade %** sliders as a

percentage value. The switch **use res** turns on the use of the pixel values in the text edit fields.
Supported color depths: 24 bit

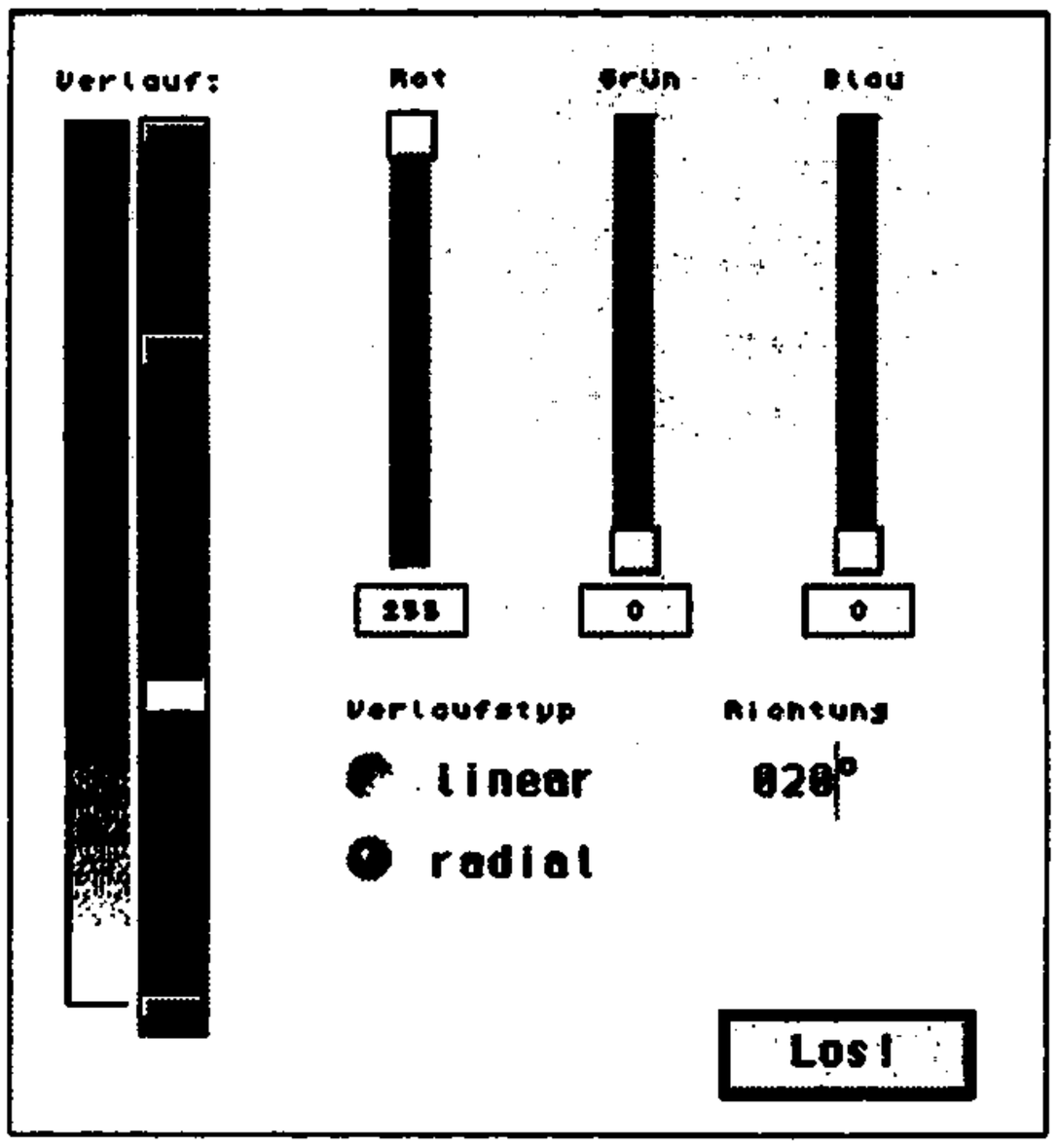
6.11 Equalizer

A little hard to explain, what this module does.
It tries to expand the image contrast, or better, tone range, to the full possible range of the color depth. The result is quite dependent on the image, so just try it out.
Supported color depths: 2-8, 24 bit

6.12 Expander

This one is easier to explain. The Expander expands the range of the brightness values contained by the image. The whole brightness range is being stretched, so that the darkest point in the image gets zero (black) and the brightest point 255 (white).
It also has to be tried out, because the result is image dependent, too, but it normally works fine on images that are too pale.
Supported color depths: 2-8, 24 bit

6.13 Color gradient



This module is for generating user defined linear and radial color runs.
The black bar on the left side is for the key points, by clicking into it, you can define up to 16 different-colored key points.
Of course, these points can be moved by dragging them, afterwards, and deleted by dragging them to the top or bottom of the black bar.
By selecting a key point and adjusting the three sliders to the right, a color can be assigned to each key point. The angle of the linear gradient can be set in the text edit field, the centre of the radial gradient can be positioned inside of the image: a crosshair will appear in the active image window, as soon as you klick on "radial". The color run will be inserted directly into the active picture and is scaled to its size.
Supported color depth: 24 bit

6.14 free 5*5 filter

Yes, this filter doesn't cost you one buck extra ;-). With this filter you can make a beautiful image totally worthless, or you may generate very funny effects - but all of that (almost) totally free.
The module looks quite complicated and the dialogue may scare you, but even if you don't want to make your own filters, you can do something with this module. Because a huge number of filters are included with Smurf - just load them.
The matrix size of 5x5 makes quite clear, that every operation makes the calculation of 5x5 pixels necessary. The middle point in the matrix is the one referring to the image pixel, that is operated on, the other matrix values refer to the neighbouring pixels.
When starting the filter, the whole matrix is "moved" pixel by pixel over the whole image, and all 5x5 neighbouring pixels are computed to one, that the centre matrix value refers to. Any matrix value has a scale factor between -99 and +99. A scaling of 0 means, that the pixel the 0 value is referring to is not being computed into the result.
All image points are now balanced by the scaling factor and added together. At the end, the result is divided by the **Div** value, which will mostly be the value of all 25 matrix values added together.
The further options:
Bias is a "zero-point threshold", moving the value 0 of the computation up or down. **Clip** clips too big or too small filter results, and **Invert** inverts the filter result before it is inserted into the image.
The slider to the right, finally, adjusts the strength, which will be used for inserting the filter result into the image.
Supported color depths: 16 and 24 bit

6.15 Gamma correction

Gamma correction changes the image on a base gamma value of 1.0 to the adjusted gamma value. Sometimes works even better than brightness/contrast, especially when channel dependent corrections have to be done.
Supported color depths: 4-8, 16 and 24 bit

6.16 Greyscale

The slightly different greyscale conversion - with a dialogue for adjustable parameters. Of course, a normal conversion to 256 shades of grey is supported as well, but additionally you can choose, if the image is going to be totally grey, grey or just a little greyish, by the **intensity** slider.

Furthermore, the number of grey shades to be generated can be typed into the text edit field (default is 256). The images original color depth is not going to be changed by the conversion! So, if you want to transform a 24 bit color image to an 8 bit grey image, better use the destructive image transformation (chapter 3.3.2).
Supported: 1-8, 16 and 24 bit.

6.17 Brightness/Contrast

The brightness and contrast of an image can be manipulated with this one. Sliders ranging from 0 to 255 for brightness and contrast can make any image fade to black or look like bleached in a snap.
Sure, values somewhere in between the extremes will be better for serious image correction, if the flash on your camera failed again or the image is totally over-exposed. And, if you don't like the contrast appearance of an image, you can correct it by the second slider. Additionally it is possible, to make the correction operate on every color channel separately, e.g. for only giving the red channel of an image a little more contrast, or similar.
Supported color depths: 1-8, 16 and 24 bit.

6.18 Highlight/Shadow

This module makes a separate de-/increase of highlighted or dark (shadowed) areas in an image possible.
To be precise: The brightness of all image pixels with brightness higher than the adjusted highlight value is set to 255, the other's brightness is being increased by a value that is computed by their original brightness. Vice versa, all image pixels with brightness below the shadow value are set to zero (dark).
Supported color depths: 2-8, 16 and 24 bit

6.19 HSV it!

It's plain but quite powerful, rather plain and you're rather helpless watching how seemingly quite slow it is.
Or something like that.
HSV it! enables you to change the colors of an image in the HSV and (notwithstanding the name) in HLS color model.
These color models are quite good for color corrections, for brightness (Luminance), Color (or Value) and Saturation can be changed independent from one another (see also Appendix A).

So, there we are, taking a look at the dialogue. Only two radio buttons, three sliders and switches. The switches (below the sliders) choose, how the values adjusted with the sliders are used for the correction, the radio buttons toggle between HSV and HLS color model. A "r" in a switch means, that the value adjusted is used in relative mode, meaning it's being added to the pixel's values. "a" means, it's an absolute (not obsolete ;-)) value. If a switch is empty, the sliders value is not going to be used at all, the referring image pixels component is left untouched.
That's almost anything that can be said. But, because you probably can not imaging what is possible with this module, we give an example:
If an image looks a little pale (happens very often with scanned images), you could use a brightness and contrast adjustment. But, a saturation correction looks more natural: toggle the H and L switches off (empty) and the saturation correction to "r"elative mode. - Now, you can increase the color saturation of the image by the "s"aturation slider.
For the color adjustment, there is no relative mode, because a relative colorizing of an image makes not quite sense. With the color correction in absolute mode and the other corrections turned off, you can easily generate duplex images in HLS color model. Well, you better try out for yourself.
Supported color depths: 1-8 and 24 bit

6.20 Invert

The most simple module inverts an image (believe it or not, without quality loss). This can be reverted by inverting the image again.
Supported color depths: 1-8, 16 and 24 bit.

6.21 transform VDI-/hardware

This module is not going to be needed very often, but then it's good to have it.
Sometimes application programmers misunderstand the GEM-Image or the True Paint Image file formats. Meaning, the palette is saved the wrong way, using the VDI color index table, not the direct color indexes. Hardware conversion will remap the palette to the correct indexes in this case.
VDI conversion is the same, but the other way round, e.g. for saving an image for such "misunderstanding" applications.

6.22 Magic Picture

This module is a contribution by Bjoern Spruck, who we want to thank very much for it.
Magic Picture computes "magic" 3D-images, like shown in the book "Magic Eye".

These images contain 3-dimensional information which is put into a texture or noise. A special way of looking at the images results in a three-dimensional impression. Looking at them a normal way will show nothing but a colored pattern. But where's the promised 3-dimensional effect, or ...

... how can I see them?
If you take a closer look at such an image you will realize that the pattern is repeated from the left to the right hand side of the image, whereas every repetition is a little different. To see the 3D-image, you look for a particular, easily visible repeating point (for example a black or white spot in the texture). Now look at the image as if you wanted to look straight through it, so that the two black or white spots next to each other move towards each other and end up at one point. If they are, you are able to see the 3D-image which looks as if it is behind the 2D-image.
If you can't make it, take the image (or the monitor) and look over the upper edge at a distant point, so that your eyes look as straight forward as possible, not focused at one point. A third way to do it is to move your head so close to the image (the screen) that your nose actually touches it. Now wait until your eyes relax and look through it. If your eyes look straight forward, move away from the image very slowly until you have reached a normal viewing distance. Don't try to focus your eyes on the image.

6.22.1 how it works

The module needs two images: the depth image and the texture. The destination image is always being overwritten by the resulting 3D-image and only gives the size of it. The computation color depth is always 24 bit.

The depth image is in grey shades and defines the 3-dimensional object with the brightness scales. The brighter a pixel is, the closer it will appear in the 3D-image. The text edit field "maximum distance" gives the range of the 3D-space from white to black color in the depth image.
The depth image can be of any size, it will be scaled when the 3D-image is computed. For a better predictability of the result it should be a greyscale image.
If you want to get a 1:1 scaling of the depth image to the 3D-image, the height of the destination image has to be the same as the height of the depth image, the destination images width has to be the with of the texture PLUS the width of the depth image.

To generate a 3D-image from a depth image you need a texture. This can be any image in any color depth and size. When computing the 3D-image, the texture will be copied

several times and scaled horizontally by a factor computed from the depth image, so that different information for each eye will be provided when looking at the image the right way.

The texture can be a random noise, a photo or whatever you want, it's only a matter of your taste. The width of the texture stripes can be adjusted by the corresponding text edit field. The maximum value here is the distance between your eyes (somewhere around 3 inches). That means that the distance from one texture repetition to the next should not be bigger than 3 inches, because you won't be able to put each eye on a different texture stripe then. However the distance should not be smaller than 1,5 inches because this may result in seeing more than one 3D-image. The texture width also is responsible for the distance of the most distant point in the 3D-image to the 2D image.

With textures showing an object clearly, an individual column width has to be given for each column to prevent sharp borders where one column touches the other.
Mostly the computation modes O>>>, <<<O, <O>>, <<O>, >>>> or <<<< ar the best for this kind of textures. Not any computation mode is good for every depth image or every texture. You may have to try it out.

6.22.2 module configuration

computation mode:

Using different direction and mode of computation is good when you want to create particular effects. Normally the image is computed with its center as the origin of space (<|>> and <<|>). This means, the texture is almost kept original at the center of the depth image, and distorted to the image borders. >>>>, <<<<, |>>> and <<<| mean, the texture is undistorted at one of the image borders.

maximum height:

This is the complete Z-range of the 3D-space in which the image is computed, meaning the bigger this value is, the more distant the darkest point of the depth image will look in the 3D-image. Values from 5% to15% will have good results. If this value is too big, the texture may be distorted too much and no 3D-effect will be visible.

mirror/tile:

If the texture height is smaller than the height of the destination image, it has to be repeated to fill the size of the destination image. "Mirror" will mirror every other repetition.

6.23 Maximum/Minimum

It's not that this module couldn't decide what it wanted to be. It rather helps images finding a totally new way of thinking about themselves. Hmmm ...?
Alright, alright. In an adjustable radius of each image pixel, all pixel's brightness values are set to the maximum/minimum brightness found in the radius.
Supported color depths: 8, 16 and 24 bit

6.24 Noise

Don't worry, it's not getting very loud.
We mean video noise. You can choose grey or color noise, whereas the color noise can be adjusted for each color channel separately - bad image quality by a keypress.
Supported color depths: 24 bit

6.25 NTSC

Yes, the american TV standard. Apart of the ... well, strange, color mapping that's said to be the reason for the name ("Never The Same Color"), the NTSC standard has another special. Every second line on a NTSC TV looks darkened. The reason for this has to be something like the reduction of flickering or something like that.
OK, this module just darkens every other line of an image by the adjusted percentage.
Supported color depths: 16 and 24 bit

6.26 Painting

Smurf is an artist.
Painting simulates the look of an oil painting, works best on photographed images.
Supported color depths: 8 and 24 bit

6.27 Scatman's World

Scatadubelibbebielelidau!
A little homage to Scatman John who died to early - his great char hit from 1995 quasi as remix-module, brought to you by Smurf. But he was the reals scatman.
This module "scatters" all pixels in an adjustable radius randomly, with also adjustable strength - think of this effect what you want, we like it. ;-)
Clear allocates a separate memory block for the destination image - this will produce a "clearer" result, but needs more memory.
Supported color depths: 8, 16 and 24 bit

6.28 Shear

This one shears images by a free adjustable angle on both axis.
It's like painting a picture on a rubber sheet and pulling on two opposite corners.
By shearing on both, the Y and the X axis, a 3-dimensional effect can be simulated.
The **interpolate** switch toggles between a non-interpolating but quick mode and a slower but more accurate mode with higher quality, which will be visible especially on the image edges.
Supported color depths: 8, 16 and 24 bit

6.29 Sinus waves

Now you can do what is forbidden when driving a car: wavy lines. Ups and downs - no problem with this module.
A sinus wave distortion can be applied to an image on the X (horizontal sine) or the Y axis (vertical sine). Two sliders adjust the sine parameters.
Strength is the amplitude of the waves, **frequency** the distance from one peak to another, both in pixels.
The **keep size** switch refers to the image size. Normally, the image will be increased by 2 times the amplitude of the wave. Selecting the check box will tile the image borders, mapping an image edge to the opposite edge when distorting. That keeps tileable images tileable after the distortion.
Supported color depths: 8 and 24 bit.

6.30 Scale

Is an image too tiny or far too huge for you?
Is your coffee too cold or the day too rainy?
Doesn't matter, because at least for the first two problems we have a solution. Free scaling of images is possible with this module.
A scaling factor (percentage values) can be adjusted by the two sliders, alternatively a destination image size in pixels can be typed into the two text edit fields. When one of the edit fields is left empty, the corresponding axis is being scaled by the original aspect ratio of the image.
One problem when scaling digital images is always, that pixels disappear when scaling to a smaller size, and the pixel doubling when increasing the size. But there is also a solution for this problem - almost. The full detail cannot be kept, but at least the quality loss can be reduced.
The magic words are interpolation and extrapolation.

Interpolating means, that when increasing the size of an image, a pixel is not doubled but a linear interpolation takes place from one image pixel to another. The same for reducing the size, here pixel rows and lines do not disappear rather than being taken into the neighbouring pixels.
With the **interpolate** switch turned on, these techniques are used. The image has to be converted to 24 bit for successful high quality interpolation, meaning an additional memory block in size of the image with 24 bits color depth is needed.

The destination size can either be adjusted by the percentage sliders (ranging from 50% to 200%) or by the text edit fields **X res** and **Y res**, in pixels. When the check box **use res** is selected, the pixel values from the edit fields is used, otherwise the percentage sliders.
If you want to give scaling factors only for the x axis of the image and constrain the original proportions of the image, you can just empty one of the edit fields by pressing escape.
Supported color depths: 1-8 and 24 bit. Interpolation converts the image to 24 bits.

6.31 Solarize

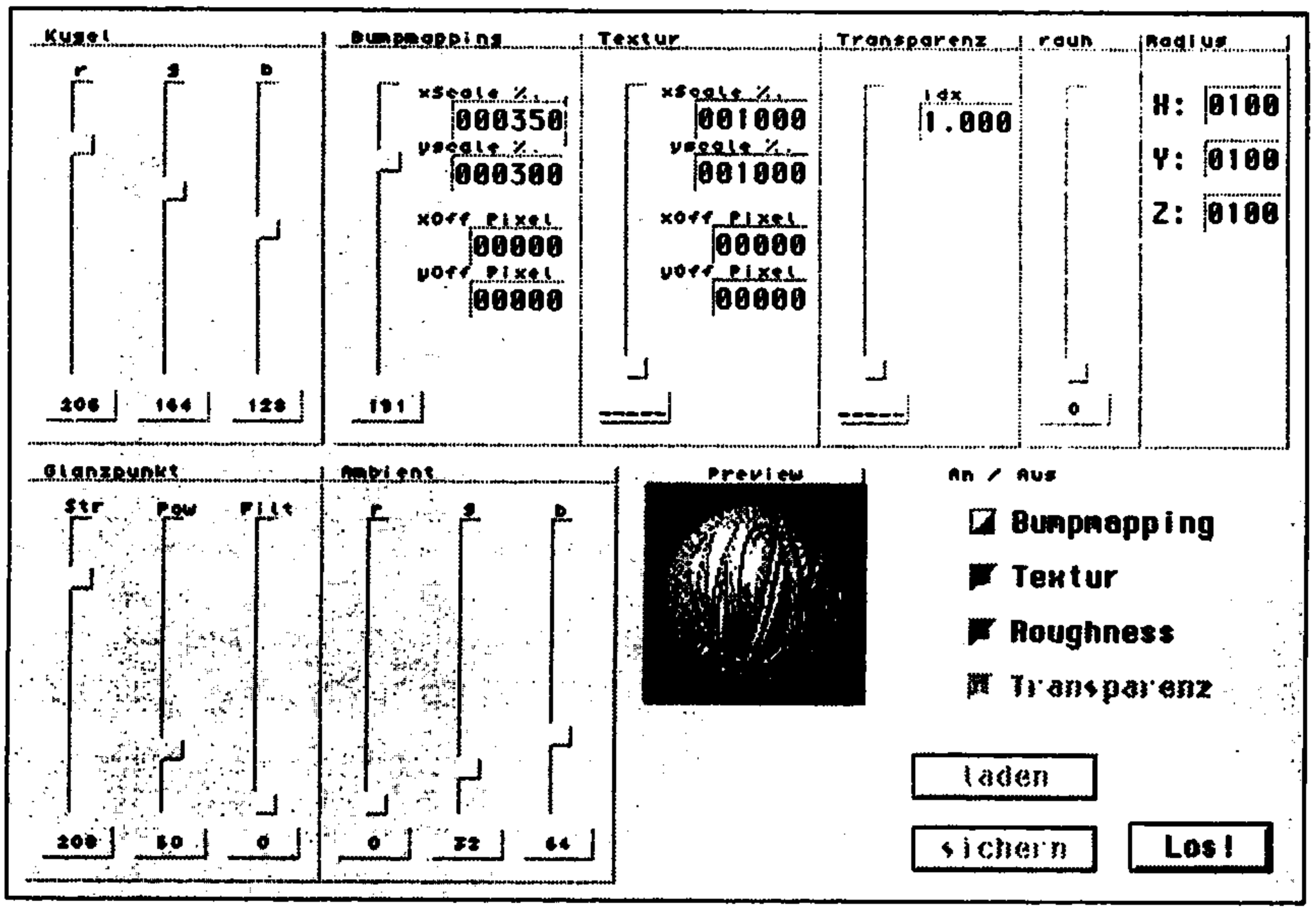
This module simulates the effect of exposing a photo to light while it is being developed.
Supported color depths: 1-8, 16 and 24 bit

6.32 Speed-O-Move

This one simulates a motion blur, like if you photographed a fast driving car with high imposition time.
The two sliders adjust the strength of blurring (corresponding to the speed of the car you want to take a picture of) and the direction (angle).
Supported color depths: 16 and 24 bit

6.33 Spherical Image

This module is for your balls. Erm, for generating balls for you. Er ... forget that.
Even, if you don't find the dialogue as easy to use as a ball pen - flexibility and a wide range of possibilities do not fall from the skies.



For the beginning, just take a look at the three **Sphere** sliders. These adjust the color of the sphere's surface. When you now klick on the preview box, a lovely sphere is being generated already.
If the size is a little too little, you can type in the radius at the **Radius** text edit fields (you won't see that in the preview, it's only for the final image).
The next you can do, is play around a little with the **Ambient** sliders, which are for the color of the ambient light.
The **Specular** sliders are for **brightness** and **power** of the specular light on the surface of the sphere. The **filter** slider filters the specular light with the surface color of the sphere, which will make it look metallic.
A further simple slider is the **rough** slider. The roughness of the sphere surface can be adjusted with this one, 0 means shiny and 255 matte. The **roughness** Checkbox has to be selected, otherwise the roughness slider will be ignored.

It's getting little more difficult with the **Bump mapping**. A bump map is just another image, which is being "stamped" into the spheres surface (you may remember it from the **Bump It Up** module). The image you want to be used as a bump map has to be as-

signed to the module by dragging it from the pic manager onto the Spherical image module. Choose **bump map** in the popup that will appear.
The text edit fields **XScale %.** and **YScale %.** are for the horizontal and vertical scaling of the bump map. The slider in the bump mapping parameter box adjusts the strength of the bumps on the sphere surface.
These values are used inverse, meaning the bigger the value, the smaller the bump map. For using the bump map, the **Bump mapping** checkbox has to be selected.

The options for texture and transparency are only available from version 1.0 on, but we'll explain it here for future versions of the module.
The texture is different from the bump mapping. It will be wrapped around the sphere, like the bump map is, but it only changes the surface colors, not the structure. The Offset and scaling sliders are the same as in the bump mapping parameters.
Finally, the transparency adjusts, how transparent the sphere is going to be. But this is not only a filtering of the sphere onto the image, the transparency is refractive, meaning, it simulates real transparent materials like glass, including the distortion when you look through it. That's what the **index** is for. This means the refraction index and is corresponding to the physical density of the simulated material: 1.5 is the refraction index of glass, 1.8 is the diamond index, 1.4 is like transparent plastic.
The index influences, how much the direction of light particles (photons) is changed when moving e.g. from the air into the material (the transition from one to another material with a higher density will cause the photon to change its flight direction, this happens because of the wave characteristics of light - see physics literature for a closer explanation) - a process called refraction. The refraction index is a multiplier for the angle between the light direction vector and the surface normal. Thus, an index of 1.0 means, the light is not refracted at all.
For using the transparency feature, the check box **transparency** has to be selected.
Finally, the position of the centre of the sphere to be generated, can be set by the crosshair which will appear in the destination image, as soon as you have assigned it to the module by dragging it from the pic manager.
Just in case you wonder, that the module does not render a preview immediately after parameter changing: this happens only when clicking the preview box, because the computation is kinda time intensive.
The module looks quite complicated, and that's what it is. We can just recommend to play around a little with it, to get a feeling for the parameters.
Supported color depths: 24 bit destination image and texture, 8 bit grey bump map

6.34 Mirror

Mirrors an image vertically and/or horizontally. 2-axis-mirroring and spinning by 180 degrees have the same result, the difference is only the algorithm. Mirroring will need significantly less memory than rotating, but it's a little slower.
Supported color depths: 1-8, 16 and 24 bit.

6.35 Twirl

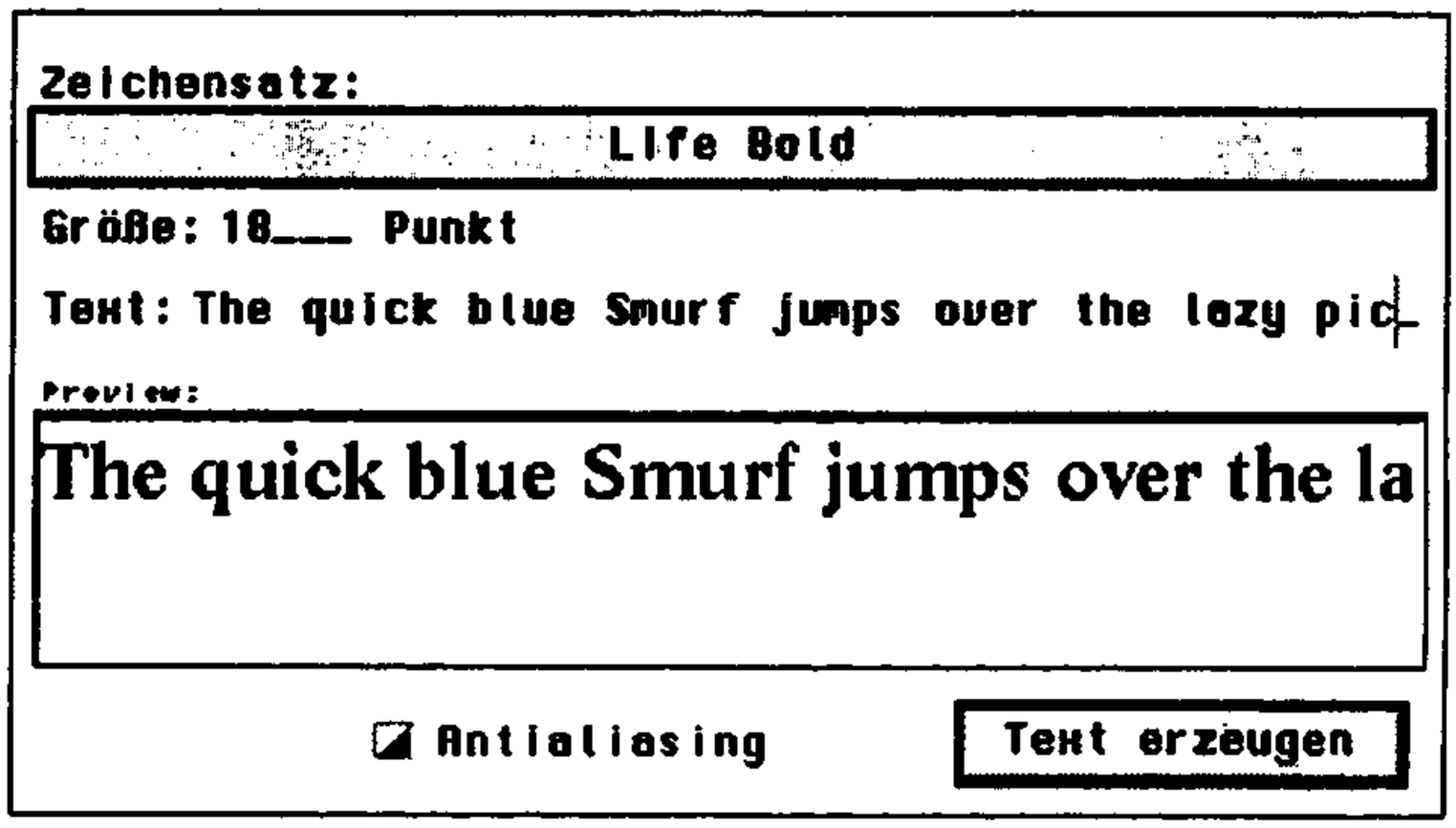
You actually could throw your pictures into the sink and pull out the plug, but then it'd be wet. So rather forget this, let this module do the work and enjoy the (dry) result. You can choose the angle of twirling and two further options. **Borders** will include the image borders into the twirling. **Tunnel** will generate a tunnel effect by darkening the image to the twirl centre.
Supported color depths: 24 bit

6.36 s/w-Threshold

This conversion of any image to black and white doesn't dither. All pixels with a brightness above threshold adjustable by the slider are set to white, all pixels below black. The results may remind of the cliparts from the Beginning of the ST.
Supported color depths: 1-8, 16 and 24 bit.

6.37 Text

With this one, you can insert graphical Text into your images, if you have NVDI (at least Version 4) or a equivalent GDOS.



The module supports external XFSL-font selectors, e.g. Calvino or XUFSL or the MagiC font selector. Apart of that, the font protocol is supported, so that you can drag a font from a D&D font selector to the module. The text will not be inserted directly into the image, but into a new block inside of the active picture. This enables you to position the text and insert it with one of the block modes. A block eventually already in the image is abandoned!

Antialiasing will generate smooth edges. The text is antialiased to white color, so that antialiasing to an image background can easily be done by using the multiplication mode when inserting the block.
Supported color depths: 1-8, 16 and 24 bit.

6.38 Soften

Help! Your photomontage is too amateurish, your dog's fur looks much too bristly on the photo, or there are just too many disagreeable details?
This module was made for that kind of problems. Pressing a button will let you forget any fabric softener.
The slider adjusts the strength of blurring.
Supported color depths: 16 and 24 bit

6.39 Weight of Color

A quite simple module. The three sliders correspond to the RGB color channels, whose intensity will be in- or decreased by the adjusted percentage.
Supported color depths: 1-8, 16 and 24 bit

6.40 Wind

If you've ever been hanging up photos in your garden for drying them and, in spite of the forecast, strong winds or even a storm came up, you may know what expects you.
If not, this module will help you to imagine the results, if you did.
And, in contradiction to all weather forecasts, you can choose the direction (from the left or right) and strength of the effect.
Supported color depths: 24 bit

6.41 Zoom

Another module by Björn Spruck, and again thanks for it.
Zoom generates an effect that looks like moving the camera very fast towards or around an object when imposing the picture.
tangetial will make the picture look as if the camera was rotated very fast during the imposition time.
negative will result in a different effect that will make the image blurring exactly vice versa than it would look when it was done by a real camera.-)
Supported color depths: 24 bit

7 Chef de protocol

"Please take this off the records."

Almost every judge in almost every TV series about lawyers.

Nowadays some standards asserted themselves also on our platform, which any program (as far as it makes sense) should support. In our opinion, this makes the user's work a lot easier and the application better integrateable into individual environments. Thus, Smurf supports some of the most popular protocols, in particular these are Drag&Drop, Font protocol, AV/VA-protocol, xFSL and OLGA.

7.1 Atari Drag&Drop

The Drag&Drop mechanism makes data exchange between different applications possible in an easy way.
For example, it is possible to drag a text block you want to delete from your word processor onto the trash can icon of the desktop, and it is going to be deleted. Or, you drag data from your database to a text editors window, which will then insert the data into the text.
Smurf uses interprocess-D&D for loading and saving images and for deleting and inserting images and blocks. When dragging an object onto a dialogue window of Smurf, it is taken as a new image. If the Drag&Drop target is a Smurf image window, the transferred data is inserted as a block into the image.
This way, you can e.g. drag an image from your image database to Smurf and edit it there - without saving, program switching and loading.
It also works the other way round. Just drag a block frame or a whole image (from the pic manager) and drop it over your databases window, which will then insert it there. Of course only, if the database application supports the protocol.
Up a few lines I wrote about interprocess-D&D, because you can also use Drag&Drop operations in Smurf, without other applications being involved (e.g. when you assign an image to a module). This may look the same, but has got nothing to do with the D&D protocol specified by Atari, because there is no communication between two different processes.

7.2 Font protocol

This simple protocol makes it possible to drag a font style, size and color from a font selector to an application, which will then take the font specifications and apply them. This is done, for example, by our text module (see 6.2).

7.3 xFSL

This one was developed as a substitute for the UFSL font selector protocol, which grew quite inhomogeneous with its many revisions.
Actually, a font selector protocol was necessary, because a font selector was not originally planned in the operating system. But, with more and more acceptance of the GDOS programs Speedo and NVDI a huge number of Fonts were provided, which wanted to be selected.
In the meantime, xFSL has, even though not undisputed, made its way as a standard in the newer font selectors - Calvino, HuGo! and FontSel are three quite good examples. Smurf uses exclusively xFSL for controlling font selectors, for reasons of easy implementation and to strengthen a good standard by supporting it.

7.4 AV-/VA-Protocol

This very wide spread protocol was designed some years ago for communication of the Desktop Gemini (known from the Geneva operating system), the "Venus", with desk accessories. Developed in a time, when only one program could run at a time, it is still of great use for communication between applications and an AV-Server (mostly the desktop).
The use for smurf is getting clear when you, even though smurf is already started, drag another image icon from your desktop to the Smurf icon on your desktop.
Smurf is not going to be started a second time, but the desktop tells Smurf to load the image by an AES message. Although Ataris D&D-protocol is much more powerful and flexible, the AV protocol also makes easy exchange of objects between application and desktop possible. When dragging a block from Smurf to the desktops trash can, the block will be removed. Or, if the destination of the drag operation is a directory window or a folder, the object is going to be saved at the path specified by the destination.

Same for putting images or icons to the clipboard, when a clipboard icon is on the desktop.

7.5 OLGA

This is Smurfs latest love. OLGAs full name is Object Linking for GEM Applications and her job is the management of automatic interaction between different applications. Example: If a DTP program displays a document containing a bitmap image, and this bitmap - in a multitasking environment - is changed and saved from another application, a raster graphics program, and both programs would run at the same time, the DTP application would update its display of the bitmap after the change, automatically.
In the meantime, OLGA is used by a wide range of applications, such as ArtWorx, CAB, Everest, IdeaList, Kandinsky, Papyrus, PixArt, STELLA, Texel, and of course Smurf.
OLGA is based on a client-server-architecture, meaning, one application provides objects which are used by the others. Some applications are only Client (object user), some only server (object provider), some are both.
Smurf is only server currently, but this will change soon.
The current OLGA version, documentations and tools can be downloaded from the authors (Thomas Much) homepage: http://thmuch.home.pages.de.
For reasons of space, we can't supply the Smurf installation with an OLGA archive.

7.6 BubbleGEM-Help

Smurf supports a bubble help via the program BubbleGEM by Thomas Much.
BubbleGEM must be installed for this purpose (e.g. in the MagiC Start-folder) and it has to run for use of the bubble help - thus, a multitasking environment is required.
The bubble help is activated by the right mouse button. Klick with the right button on an object in a window, and, as far as the appropriate help file can be found, a speech bubble will open and display the help text.
BubbleGEM is also a project by Thomas Much. The current version is provided with Smurf, further informations and updates can be found on the author's home page and in various BBS systems, mostly the name of the archive will be BUBBLE??.ZIP, where ?? is the version number.

8 Color reduction

"... It was black. Where it wasn't black you were inclined to wish that it was, because the colors with which some of the unspeakable details were picked out ranged horribly across the whole spectrum of eye-defying colors from Ultra Violent to Infra Dead, taking in Liver Purple, Loathsome Lilac, Matter Yellow, Burnt hombre and Gan Green on the way."

Douglas Adams

Smurf implements various algorithms for color reduction. These methods mainly differ by the quality of the result and the CPU time consumption, say, performance.
Basically, there are two purposes for the color reduction.

- for image display on screen
What do you do, if you want to display an image with 16.7 millions of colors in a screen mode with 256 colors?

- destructive reduction
What do you do, if you loaded a 15 bit image with 15.879 colors and want to export it as a GIF image, whereas the problem is, that GIF supports a maximum of 256 colors?

The answers should be quite clear: The number of colors has to be reduced.

The color reduction method for the first scenario can be selected in the display configuration dialogue (menu Display/Configuration), for the second scenario in the image transformation dialogue (menu Edit/Transform image) or in the export dialogue.
Spreading the selections on several dialogues was not done to confuse you, or because we wanted to have more work to do. It was done to increase the flexibility and transparency of our program. Closer information to the dialogues can be found where the menu and dialogues are explained. Here, you'll get more background information.

The complete color reduction process is split in two parts: finding a palette, in which x destination colors fit as good as possible to y source colors (mostly y much > x), and the dithering which is mostly done by spreading calculation errors. Don't worry, we don't want to make your computer crash, it's just how some dithering methods work.

8.1 Palette search

8.1.1 file palette

How to load a palette from file is described at the dialogues explanation.
Normally, this method for searching a palette (well, "searching" is actually wrong, because there is no need for searching a palette which is loaded from a file. Except, if you can't find the file, then you'll have to search it, but this has nothing to do with search algorithms. ;-)) will have results with low to medium quality, because the palette you load mostly will not fit to any image (for a better understanding take a look on the Netscape dithering).

8.1.2 Median Cut

Median Cut is one of the algorithms which result in the best quality. The other is Octree, but this is not yet implemented in Smurf.
Median Cut bases on an idea from Paul Heckbert and is actually called "three dimensional color quantisation". It tries to find a number of colors out of a bigger number of colors (mostly 16 or 256 out of 16.7 millions) and to make each of the colors found, cover a as big as possible color space from the source color space.
For this purpose, a histogram is generated. This is a kind of frequency table, in which the frequency of each color of the source image is being counted.
This histogram is generated as a three-dimensional cube, each dimension represented by one color component from the RGB color model. Now, the algorithm cuts the cube a few times (sounds brutal, but it isn't so bad), so that a number of small boxes remain, the same as the number of colors you want to reduce to. Then an average value of the colors within each of these boxes is computed, which is a color representing one of the destination colors (if you didn't understand this the first time, don't worry - neither did I. And I implemented the algorithm. - The translator.)

When Median Cut configuration for screen display is set, it will not perform median cut on images with a color depth equal or smaller than the screen color depths. In this case, the system palette is just set to the image palette. This is the fastest and best in this case.

8.2 Dither routines

"Ok, you have do make the failure before you can see it!"
Olli while explaining the Floyd-Steinberg-principle to Dale

8.2.1 Nearest Color

I have to admit, the Nearest Color algorithm actually is no dither algorithm, because it doesn't dither. And this is the reason, why it's the worst looking of all algorithms (meaning the result, not the algorithm itself). But this simplicity comes along with very high speed of color reduction, especially to system palette.

This method runs through all pixels of the image and searches the nearest (closest) color in the destination palette for each pixel. "Nearest" refers to the distance in the color cube, mostly RGB. The advantage of this method is a quite high speed. Disadvantages are the loss of smooth color gradients, loss of contrast and overall rather bad display quality, sometimes kinda curious looking results may be the case.

8.2.2 Ordered Dither

This is a method, that works with a special palette (except, of course, in monochrome mode). System palette dithering makes no sense here, the results don't look quite satisfying.
Ordered dither has good results in contrast, brightness and color, but it looks really good in at least 256 colors. The results normally are not quite as good as with an error diffusion method, though the dithering looks more homogeneous.

8.2.3 Floyd-Steinberg

The original and probably best variation of error diffusion, already developed in 1973 by Robert W. Floyd and Louis Steinberg.
A more complex algorithm than Nearest Color or Ordered Dither, but it also produces much better results.
The FS also searches the nearest color to the destination palette for each image pixel, but the error that results from the color reduction (the difference between the original and the found nearest color value) is not just thrown away.
The error is diffused, meaning computed into the pixels to the right, below, right below and left below the computed pixel.
Probably this is the best method of color reduction. It produces a quite homogeneous raster with high quality in contrast, brightness and color. Floyd Steinberg dithering takes a lot of time, so we recommend this method for final image saving, when image processing has already taken place.

8.2.4 Fast Diffusion

Basically very similar to the Floyd Steinberg method, only the error is not spread to four but to one pixel. The dithering is quite fast and the results have a good quality in color, contrast and brightness. The disadvantages are, that the dithering may result in vertical lines and the raster is not quite homogeneous.
We recommend this method for image display on screen, because it's a really good compromise between quality and speed.

8.2.5 Greyscale

This one converts the images to as many shades of grey, as colors are available in the destination color depth. This means, only the brightness information will be used for displaying the image, and all color information will be lost. The results look quite good even in 16 colors (grey shades).

9 Graphics formats

"What the heck is Kontron IBAS?"

Dale

It's a true miracle, how many different image file formats exist, and always new ones come up, like flowers on a wide plain in the spring - or rather like salesmen knocking on your door on sunday morning, when you want to sleep. The huge number of formats is hard to be managed.

After GIF, there is PNG, Corel say, their CDR is standard, Adobe Illustrator format is called EPS but it is not really, PICT is quite unknown on non-Mac-platforms, TIFF can have thousands of sub-formats, Degas is totally out of fashion anyway, and my name is Olaf.

In spite to these facts, we tried to handle as many image file formats as possible (and this is quite good, cause this is the task of a graphics file converter, somehow).

Earlier, here was a long list of the supported formats. But, because this list changes every few days anyways, and just wastes space, this list will only be provided as an ASCII File on the disk.

We think, you can live with it - if not, tell us.

We don't actually want to isolate and take all the fame (if there will be any) for ourselves. Smurf was developed with the intention to solve problems, the problems of anyone (as far as problems with image file formats are concerned).

And this is why anyone can program a import module and release it or not (we'd like the first possibility much more, and the users will probably, too).

If you want to know, how to program a Smurf module, ask us. You can get a full documentation on disk or paper. We'd be happy, if you want to make it a freeware module, if you send us a review version, before releasing it. We're going to check it intensively to grant a continuous quality of all modules. If you want, we can also include modules in the Smurf distribution.

10 what Smurf can't do - on purpose

"Nobody is Prefect, except Ford"

A feature known from other applications, like Graftool of GEMView, is the generation of an image catalogue.
This is nice, but we didn't implement such a thing in Smurf. We think, this feature should be left to a specialist.
The program of our choice is called STELLA and is an image database by Thomas Künneth. Of course, image catalogues are not its only feature, it can do (almost) everything you'd expect from a database, including a few image processing functions.
STELLA is shareware and available in several mailboxes, on the internet, or you can get it from the author or from us.

Smurf cannot print colored on color printers.
Er, no, this is not totally true. Smurf can, but NVDI 4 or newer has to be installed.
Because here, too, we want to rely on an expert in this concerns.
NVDI (from version 4.1x) does not only provide the well known graphics accelerator, TrueType and Speedo for the system, but also GDOS printer drivers (and by this systemglobal) for nearly any printer, including rasterization and color calibration.
You can get NVDI from Application Systems Heidelberg, from Behne&Behne Software or from us.

Smurf is not able of making coffee. Although many users of various programs want this feature, it cannot be found anywhere.
Well, it's not that we don't want to implement it. But, do you know a coffee machine with serial port? Well, you see?
You know one? Bring it here, we'll talk to it.

11 Suggestions

Any software is living on the users.
This is not only a kind of saying we took from other documentations, but our true certainty.
If you have any suggestions, questions or wishes concerning Smurf, please tell us. We can be reached at the addresses below per post, telephone, e-mail, per fax or personally.

We don't include a formular here, because we know from our own experience, that it wouldn't be used. But what should you tell us?

Bugs:
Your system configuration (hardware, Operating system, installed resident software, free memory), the Smurf version, and an error description as detailed as possible.

Suggestions and wishes:
The suggestion and, if it's about a missing file format, some example images (or where we can get some) and, if possible, a format description (or where we can get one).

We wouldn't like to be taken for so stupid, that this has to be remarked. But this can only be of use for you, because we're going to be able to answer you more accurate and faster.

Threatenings or destructive criticism doesn't reach us at all.

12 Updates and prices

"How much expensive does the price cost?"

Trader from the DSA game

Upgrades will take place when we think it's time for it. That means, when some bigger new features have been added.
The price is going to be announced with the upgrade announcement and will be in acceptable range. Normally, upgrades over more than one version number are possible, so you don't have to buy any upgrade, and the price is going to be one, no matter from which version you upgrade.

For we're constantly developing Smurf, theoretically every day a new version could be released (we don't do this, of course). If you want to get the newest version, e.g. when we announced one, you can get it **only from us** by sending us a disk and envelope with stamps for our postal expenses and your serial number. Of course only, if you're a registered user.

Upgrades are available from us or at your dealer. For both, update and upgrade, we need the filled registration card. If you did not get a registration card with your Smurf, please write us.

We hope, that we don't have to remark it, but we do it anyways.
Anyone who gives away unauthorised copies of Smurf without our explicit, written permission, or who uses unauthorised copies of Smurf, will be confronted with legal steps from our side. Same for modules, resource file or any other part of Smurf, including the manual.
An unauthorised copy is any copy of the program or any of the program files, or the manual, that has not been authorised by us with a written and signed permission. This includes copies of the specified data to internet or to mailboxes.

13 we about us

"By the way, from funny to crazy it's less far than from genius to insanity."
unknown

After all of these pages we wrote about "I", "us" and "we", the question may come up: "who is it"?

We are Olaf Piesche and Christian Eyrich. We're part of Therapy, the good name with the bad Falcon-demos (but nevertheless running under MiNT).

Therapy (without addition) is existent since August 1994, since the famous living room-convention took place. Well, we had a different name at this point (Olaf was called Michael and I was called Peter, no, of course I meant us as a group, when I said "we", not every single person). However, we wanted to make Falcon-demos.
Time jump
Eastern '95: The Fried bits III brought our demo "function main()" the fifth place in the competition (of five), even though there was a quite cool fire effect. It brought me a lot of sleep, Micha ... er, Olaf, a bad cold and the whole group new friends and apart of that, unrecognised by the creator, the name for the next demo.
Time jump
11. sept. '95: The Drax-convention, and besides that (unrecognised from almost anyone) the day, at which Smurf was announced officially. Basically, it was Dales idea and resulted from the frustration of having soooo many file formats which nobody can read, and if they can be read, to need two cups of coffee time to do it.
Time jump
Eastern '96: Symposium (or Fried-bits IV, which is not quite familiar because of the, in relation to the real FB, bad organisation) brought our demo "Wurscht wie der Ordner heißt" (german play on words, should we explain it? E-mail us.) the first place in the competition (even though, or because there was no fire effect in it), Dale two cigarettes in his cola and Olaf a cold again.
Time hop
In the meantime of this events and today, when we are selling Smurf, there were three preview versions. The first was running on almost quite no computer, the second on many, and the third which was released on the '97 Atari show in Neuss, was running quite stable.
Some people even wanted to have a full version there, but we thought, it wasn't good enough for a full release.

Actually, there is no work share between us, we all do the same things:
One programs something, and he likes it, and that's why the other one doesn't like it and deletes it. All this again vice versa and from the beginning.
This can lead to the conclusion, that Smurf only contains the things, that each of us could hide from the other. This is basically the reason, for the low size of Smurf.
There is another reason, logically resulting from what I wrote - who finds it, will get one Smurf free.
Well, the work is split somehow. Olaf makes the main part with all the GEM stuff and half of the edit modules. Dale, when he was with us, wrote half of the import modules, the first Median Cut and the first FS dither. And I myself write the other half of the importers, exporters and edit modules, two dither modules and sometimes i bungle into Olaf's work.
As a master of the written word (how funny ... actually you can't imagine my wonderful work - now you only know how bad Olaf's englisch is. ;-))) and the writing mistakes (how true), that pleasure your eye through the whole manual, I may be allowed to announce myself, now. :)

OK, that's us, when we're together.
There are single therapists, in the following you may take a little look on who really did the Smurf thing. Who of you really can't stand this, is allowed to skip the next 10 pages and continue there (right in the midst of the appendix).

Olaf
When I was born on the 7th of september in 1973, the world didn't really know, what was coming up to it. Now, that I am developing Smurf with the one who has written till now (see below), the world does really know it, and wishes, it had already run away much earlier.
I did my first tries in programming at the age of eight on a Siemens PG670 (with CPM, brrr ...). My first own computer was, years later, a Commodore 16 with Datasette, which I expanded after half a year to the unbelievable amount of 64 kilobytes memory. Jealous on my brother's ST, I bought one, too, which I then tortured with GFA-Basic. When the ST wasn't enough anymore after three more years, I bought a Falcon030, which I still have and I'm happy and satisfied with, and programmed tools, that the world didn't need. Then I was taught C by my brother and taught it myself. Just, when I made my first C steps (I was rather stepping around on one place than forward ...), I met Dale at school, he bought a Falcon too, and we programmed like insanes and had those great plans anyone starting to program has. Christian and I met by some strange incidents, that are related to Andras Kavalecz, a little software company in the middle of

the town Schwabach and other things, with which I don't want to bother you, because they may ... well, bother you.

Christian
Christian's name is actually Christian, and he just calls himself like that.
At the moment he is 24 years old, which can be concluded to a birth date around the 13.11.1974. Apart of that, he hates to write about himself as if he was someone else, and that's why he'll stop it now.
My first computer wasn't one, but a ATARI 1040STF with SM124 - we write the year 1988. After I didn't feel like I could handle the included Metacomco Basic, I changed to OMIKRON Basic 3.0 and had the desire to program something truly great (how boring). First, the source codes were lost, and I had to start over again, then there were graphics cards and multitasking systems (by the way, I'm already on the Mega STE, which I bought at the end of the year '92) and the Basic didn't feel like it could handle the MSTE, so I lost it.
At the first of september of the next year I sneaked to Vobis (yes, I admit it) and bought my first Modem - 2400 bps. Since then I mostly run through the MausNet, tell something to the whole world (meanwhile with 14400 bps) and nobody listens to me.
In december, C came over me (or did I come over C?), then the TT and with the coincidence of May '94 my first convention - this is meant to be the breaking point in all our lives, and basically the reason for Smurf, too, because I met Olaf and NoHow there - and in April '95 the Falcon.

lyrics, voice, er, support:
Olaf Piesche
unknown
also unknown
Tel.: you know what it is?
e-mail: olaf@therapy-seriouz.de

background, FX, er, orders:
Christian Eyrich
Verdistraße 28
D-90455 Nürnberg
Tel.: +49-(0)9122/75301
e-mail: christian@therapy-seriouz.de

We also have our own page on the net, the internet. The URL
http://www.therapy-seriouz.de will lead you to news, hints and our bug database. Demo versions, updates and new or changed modules can also be downloaded there.

14 epilogue

"Some time I'm going to get them - even if it's the last thing I do in my life!"

Gargamel

If you, dear reader, have made it to this point (and didn't cheat), we can congratulate you for that.

Eventual mistakes and grammatical stumble-blocks, that you may have found lying on your way, were placed there with great care and have to be left where they are.

We also want to remark that you, if you have the opinion, that only totally insanes can have created this, are, with the biggest probability, right.

We want to thank (in no special order) the following persons:

- all people who stood around us during the development of Smurf, and asked themselves (and us), what we're actually doing there, and to whom we've given not enough of our time
- Matthias Bracke, Christoph Bradel, Thomas Künneth, Hans-Joachim Riedl, Rainer Wiesenfeller as well as their computers and graphics cards for testing, criticising and consulting
- all the people who motivated us during the development of Smurf with their constructive advice, suggestions and bug reports
- Klaus Holtorf and his drunken type setter for the nice book
- Dieter Fiebelkorn and ATARI.Programmieren for their hints, as broad as they were
- J.D.Murray and W. VanRyper for their encyclopedia
- R.O.M. logicware for papyrus
- Karim Saleh for the repro of the manuals from the second german edition on
- the world, for being there
- Douglas Adams for his four-part trilogy in five tomes
- the Running Design Team for Running, the DOOM-clone that helped us relax
- and of course all developers of graphics file formats all around the world, who made our work necessary with their exertion to make the jungle even more dense, and gave us many hours of work.

Appendix A - a users guide to the graphical terms

As I said, we tried to create a roughly understandable manual, but we did not want to do without foreign words in the main part. So we want to explain several connections and terms here.

Basically, it is being differed between "dotted" and "lined" graphics. Technical terms for the first are bitmap or pixel graphics. The other is known as vector graphics.

From now on, the word pixel will be used. This is how the smallest image contents of bitmaps are called. Pixel is an artificial word for picture element. Every pixel is defined by its horizontal and vertical position in the image and by its color.

Vectors are mathematical constructions, that are, mathematically, defined by their coordinates and their direction. In graphics, there are also attributes like thickness and color that define such an object.
Because you don't have to describe a line by hundreds of single pixels, but only by a coordinate pair and the attributes - not to speak of the background, that has to be saved to a bitmap image file, too - vector graphics are almost every time smaller than their pixel equivalents. And this is for technical drawings or sketches and logos, which have clear lines and few elements.
But our real world is much too complex to be displayed my mathematical objects. Photorealistic images can not (or at least not very good) be displayed with vector graphics. Furthermore, the single elements would be too small, too complex, so that in this case, a vector graphics would even be much bigger than a bitmap.

Another important and very useful fact concerning vector graphics is given by their mathematical nature. Vector graphics have no fixed size. They can be stretched to any size without a loss of quality (imagine a line defined by two coordinates - no matter how far the two points are from each other, the line is still a line).

A.1 Color systems

For handling color information with computers, several color systems have been defined.

The most frequent is the RGB system (red, green and blue).
Setting all components to 100% results in white. 0% of each component is black. If all three components have the same intensity, the result is a grey shade.
There was no explicit foreign word in this paragraph (apart of the one to the left), so here is one. Because the addition of the full values of all color components result in white, the color system is called an additive color system.
Probably it's the most used system in all computer related cases, not least because it is in conformity with the CRT of the monitor: if no electrons hit the substance on its back side, it is black.

A further color system is CMY. This is for Cyan, Magenta and Yellow, which are the complementary colors to red, green and blue.
An amount of 100% of every component results in black, 0% is white. A grey scale is the result, if all components have the same intensity.
Because the absence of all colors results in white, this color system is called subtractive color system.
It is the most common system in printing related cases, not least because it is in conformity with the paper. Paper is normally white, and the colors reduce the intensity - the more color, the darker the paper. If you turn up all colors to 100%, you get almost black. Almost, because it's no perfect black rather than a dirty olive-grey-green-brown. And this is, why there is a variation of the CMY system, expanded by the K (also called B) component. K (B) is for Black and is quite useful for reducing the amount of color needed for dark parts, and for getting real black, actually.

The third of the most common color systems can also be found in Smurf and is called HSV. It's got nothing to do with sports and german football teams, but stands for Hue, Saturation and Value. The HSV system (often called HSB for Hue, Saturation and Brightness) describes color, saturation and brightness of a pixel with these 3 independent parameters.Thus, it is quite good for color choose panels, once you get used to it.
A close relative to HSV is called HLS, which means Hue, Luminance and Saturation. The difference is, that, in HSV system, 100% brightness displays the adjusted hue (H) at

full brightness, whereas in HLS this is the case at 50% luminance. 100% luminance will result in white.
Both color systems are not used in common graphics file formats.

Additionally, I want to name the next two, even though you as a user won't get in contact with them.
First, there is the YCbCr system. This tongue wrecking abbreviation stands for a color system which features luminance (Y), meaning brightness, and chrominance (CbCr), meaning color, independently. This independence of the components makes it quite useful for data reduction, for the human's eyes capability of differing color information is worse than on brightness information, and the YCbCr system makes a data reduction in the color information quite simple. Examples for this system (and data reduction) are the JFIF and PhotoCD formats.
Second, there is the YIQ system (used in the american TV standard NTSC), which you can imagine rather similar to the YCbCr system. An image file format saving the image date in YIQ system is not known to us.

A.2 Color depth

Color depths means the maximum number of colors, that can be used at a time. Be careful, don't mix it up with the number of actually used colors.

The color depths results from the number of bits, that are used for one pixel, so it's 2 ^ number of bits. Common color depths are

1 bit = 2 colors
4 bit = 16 colors
8 bit = 256 colors
15 bit = 32.768 colors
16 bit = 65.536 colors
24 bit = 16.777.216 colors

Theoretically, different color depths are possible, in some formats the use of 3 or 5 bits, for example, is possible. Such color depths are not very common.
When increasing the color depth, the amount of information per pixel gets higher. The raw image data will in any case need more memory, when the color depth increases - this is not valid for compressed images!

Displaying graphics with up to 256 colors or 8 bits, saving takes place as so-called palette images that often, that we can say every time. Exceptions are just few image file formats, e.g. when it's about saving images with 256 grey scales.
That means, the colors are not in the image memory directly. Every byte of the image data is a reference to a color table (the palette).
A pixel (byte) with the value 93 doesn't say anything about the color. 93 means, the computer has to "look up" the color at the 93rd place of the palette (this is why the palette or color table is often called "color look up table", or CLUT).

The RGB values, that can be found at this place in the palette is then the color of the pixel. For every pixel (byte) with the same value refers to the same color palette entry, one RGB color triple in the table is responsible for the color of all the pixels with the referring value. Thus, changing a color from the palette will result in color change in many pixels of the image. Just try it out - your boyfriend or girlfriend will suddenly have ugly green spots in the face, although you actually only wanted to give the brown eyes a more exotic touch ...
In most times, the colors are saved in 24 bits in the palette, 8 bits for R, G and B each.

When the color depth of an image is higher than 8 bits, the colors are mostly directly set to the image memory (screen modes >8 bits are all called "direct color modes"). The pixel values then describe the colors by themselves and are not a reference to a color table.

With a color depth of 16 bits, 5 bits are used for every color component. The remaining bit is mostly not used (if it is anyways, it is mostly used for the green component, because the human's eyes green sensitivity is the highest). This color depth is called HiColor. A color depth of 24 bit is called TrueColor.

There are also formats which support color depths up to 32 bits. This doesn't mean that they provide more colors as with 24 bits. The additional byte is sometimes used for the color intensity, sometimes for an alpha channel, bit most times it means nothing at all.

Appendix B - Short introduction of some formats

"My name is Bond - James Bond"

Sean Connery, Roger Moore, George Lazenby, Timothy Dalton and Pierce Brosnan simultaneously

BMP - Alpha Microsystems-format

A format developed by Alpha Microsystems for the use on workstations. It supports color depths up to 24 bit. A compression is supported, but not a very effective one.

BMP - OS/2 version 1.x bitmap-format

Originally developed by Microsoft for Windows 3.x. The only difference to the Windows 3.x version is a modified header. No compression, color depths up to 24 bits.

BMP - OS/2 version 2.x bitmap-format

Developed by Microsoft and IBM. Highest color depth is 24 bit. Compression is supported up to 8 bits color depth. The difference to the Windows version is, that the OS/2 2.x revision supports the saving of several images in one file.

BMP - Windows bitmap-format

And again, Microsoft is responsible. Original format, that the OS/2 versions base on.

CLP - Windows clipboard-format

Guess who? Maximum color depth is 24 bits again. It's of no importance for the saving of images. It was actually developed for data exchange by the clipboard. There are several different versions from different software developers, that have only a changed format recognition, for making data exchange to other programs impossible. PageMaker, for example, is one of these great programs.

GIF - Graphics Interchange Format

This format was developed in 1987 by the american mailbox company Compuserve. The main goal was a high compression rate. The maximum color depth is 8 bits (256 colors). More wasn't very common in these days and everything else would have been too big. In July 1989 a second version of the format was released and named GIF89a. This one was supplied with several optional extensions of the original format, such as the possibility of saving animations to one file, defining colors of the palette as "transparent" and saving additional text to the image file. Unfortunately, Compuserve decided to use the LZW algorithm for data compression. Unfortunately because 1994 the com-

pany Unisys found out, that they have the patent for this algorithm and demands a license fee for all new programs using this algorithm.

IBG - NASA PDS file-format

This format was thought out by the NASA for saving satellite images. The only supported color depth is 8 bit, only greyscale.

IMG - GEM Image file-format

Developed for the user environment GEM from Digital Research. Originally, only image with 1-8 bits were supported, and no color palette could be saved to the image files. In 1992 the Geiss brothers defined an extended IMG, the XIMG in their book "From beginner to GEM professional". The extension was the saving of color palettes in the color systems RGB, CMY, HSV and Pantone into the image files. But, this wasn't enough for some programmers, and so XIMG was extended by the capability of saving images with 24 bits color depth.

But there were two programmers at a time, who developed this extension, each without knowing of the other, and so there are two, the PixArt TrueColor XIMG and the GemView TrueColor XIMG. When everything calmed down a little, Guido Vollbeding released suggestions for extension of the IMG compression routines in 1994 and defined the TIMG format. Until now, as far as we know, nothing has been done since then, and TIMG is supported only by very few programs.

IMG - Vivid Ray-Tracer-format

Developed by Steven B. Coy for his shareware raytracer Vivid. Only 24 bit is supported, as a compression a simple run length encoding is used.

JFIF - JPEG File Interchange-Format

The format is actually called JFIF, but more known as JPEG. Although this is actually the abbreviation for the developers. The Joint Photographic Experts Group tried to define a standard for graphic files, using their own compression method. This format is very interesting, because the compression is extremely effective, but is based on the loss of information - more and more with every saving of the same file. But it's not as bad as it sounds, cause the compression takes place in the YCbCr-system (see appendix A). Apart of that, the compression quality is variable.

JPG - HSI JPEG file-format

This JPEG format by Handmade Software Inc. actually is none. A part of the image is saved as a 24 bit JFIF, the rest as 8bit GIF. It was developed for saving memory on image files with 8 bits color depth or less. Unfortunately, this doesn't seem to work like it should, because even 8 bit images are about double the size of the GIF equivalent. The use of the same file extension as the JFIF format is quite confusing.

LBM - Amiga IFF-format

This format, developed by the games company Electronic Arts is not really in place here. It's actual more like a container file, capable not only of saving graphics, but also text, sounds, etc. That makes it very complex and almost nowhere used than on Amiga. The images can be saved in color depths from 1 to 8 bit and 24 bit.
The saving of images is done for the Amiga video hardware, which can especially be seen in color depths from 5 to 7 bits. On Amiga, these images are displayed in HAM (Hold-And-Modify) mode, using a trick a little similar to the one used by Spectrum 512 on the ST. Optional RLE compression is supported.

LBM - PC IFF-format

For spreading the image further, there is also a PC version of the Amiga IFF format. Images are saved in "normal" encoding, not for HAM modes. Thus, only color depths of 1, 8 and 24 bits are possible. The PC version also supports optional RLE compression.

MAC - Mac Paint-format

This format comes, who'd have thought it, from the Macintosh platform. It was developed by Apple Computer Inc., seemingly endless time ago, because only black and white images are supported and the resolution is always 462*399 pixels.

PCD - Kodak Photo CD-format

Developed by Eastmen Kodak. Every PCD file contains one image in five different resolutions. Originally made for saving photos on Kodaks PhotoCD system, that makes saving holiday photos to CD and viewing them on the TV set possible for almost anybody. A detailed description can only be bought from Kodak for a huge amount of money. The image files have sizes around 4-6 MBytes. The format was extended later, from professional users, who wanted more than the Base*16 resolution (base is 768*512 pixels), to a maximum of Base*64. Later on, the format was extended for medical purposes (PCDmed) to a maximum of Base*256, which means a maximum resolution of 196.608 by 131.072 pixels (!) for saving Roentgen or Ultrasonic images or images from CST devices or things like that, in high resolution.

PCX - ZSoft Paintbrush

ZSoft Corp. developed this format for their graphics program Paintbrush. It's one of the widest spread formats on IBM compatible computers, for paintbrush comes with Windows. The format supports up to 24 bits. True color has been implemented not a long time ago, thus some programs are still overcharged with it.

PIx, PCx - ATARI DEGAS-format

More like a format generation. The series first consisted of PI1, PI2 and PI3, for images with 1, 2 and 4 bits uncompressed. Originally developed in 1985 by Tom Hudson for his painting program DEGAS. With the sequel DEGAS Elite, the PC1-PC3 formats for compressed files were released. Meanwhile, there are variations for the TT graphic modes up to PI9, resp. PC9. A special feature of the Elite format is, to save informations for palette animation in the file.

PNG - Portable Network Graphics

One of the very few convenient formats. The reason for developing it was basically the announcement of Unisys, that the LZW algorithm was patented by them and a license fee would have to be paid for its usage. Thus, the commercial sale of GIF im- and export routines is either illegal or very expensive. For creating a substitute for GIF, which is first of all used in the WWW, PNG (pronounced "ping") was developed. Furthermore, PNG breaks up the restriction of GIF to a maximum of 8 bits color depth, provides better flexibility, alpha channels, a better interlacing and a more sophisticated compression. Unfortunately, it didn't spread very wide yet, even though the specification version 1.0 was already released in the end of '96.

RAS - Sun Raster

The home of the Sun Raster format is the Sun workstations with the Solaris OS. It supports b/w, greyscale and color images in any color depth, and an RLE compression. It's quite often used in the Unix world.

SGI - Silicon Graphics Inc. Image File Format

Developed by Silicon Graphics Inc. for their workstations. It's one of the most popular formats in this sphere, for it's component of the SGI Image Library, which comes along with all SGI computers. Supported color depths are 8 and 24 bits.
Saving is possible with or without compression.

SPU, SPC - ATARI Spectrum 512

This format from the painting program Spectrum 512 is quite tricky. The program enables ST computers, which were only able of displaying 16 colors at a time in '85, to generate and display images with 512 colors. The compression is a simple run length encoding.

SPX - ATARI Spectrum 512

An extension of the original format by the Electronic Images guys.

TGA - TGA-Format

This format from Truevision inc. is one of the most popular formats of all. It's by the way really called TGA and not Targa. Targa is only the name of the video card developed by Truevision, which was e.g. able of making snapshots of analogue video streams. The software used for the control of the card used the TGA format. Graphics can be saved with 1, 8, 24 and 32 bits. It's so wide spread because it was one of the first formats supporting true color, so almost any application should be able of handling it, unlike PCX or BMP.

TIFF - Tag Image File-Format

This one was developed by Aldus. It's become a general purpose format, because it's extremely flexible and can be saved as a format with optimum efficiency for almost every purpose. Thus, it is also extremely complex - only the original specification (TIFF 6.0, july 3rd 1992) is 121 pages long. Some people use to call it an ideology rather than an image format. Because of that it is quite easy to make a lot of mistakes when writing TIFFs, that lead to misinterpretations by the importing programs. Even well known developers of graphics applications are not able to have a complete overview over the format.

The color depths supported are as variable as the format is (actually, even 6.5 bits in eight channels are possible). The most common used compression methods are RLE, LZW, JPEG-compression and CCIT Fax-compression for monochrome images.

TPI - Turbo Pascal graphics format

Developed by Borland for Turbo Pascal. "The saving method is hardware dependent. This leads to the problem, that, eg. an image saved on a VGA graphics card, cannot be read on a different card" says the documentation. It may be because some Ataris don't have an extra graphics card, or because of something else, but Smurf is able to read this images, though sometimes a little horizontally displaced.

TPI - Truepaint Image-Format

The TruePaint painting program's file format. Another time an example for a file format, which is neither capable of something remarkable, nor with a good compression. The data format is the same as the ST screen memory, and the data is uncompressed. A very interesting thing is the 16 bit mode of the format; it was done for the use on Falcon030, but the data is still stored in ST screen format. TPI is used among others for the DOOM clone Running for the Falcon - maybe because it's relatively simple to read and write.

VICAR 2 - Planetary File Format

The VICAR2 file format is used primarily for storing planetary image data collected by both interplanetary spacecraft and Earth-based stations. Many astronomical and astrophysical organisations use and support this format. That's why it was developed for the VAX/VMS platform and why it's supported by many UNIX image processing applications.

The number of colors usable is unlimited, no compression implemented.

WPG - WordPerfect Graphics Metafile

This format is a development of the WordPerfect Corp., developed for the use with the program WordPerfect. Saving is possible up to 8 bits. Besides the vector graphics, which is normal for metafile formats, EPS bitmap data can be stored in a WPG file.

XBM - Xbitmap

The format used for saving icons in XWindows. The data structure is a C array saved as an ASCII file, thus it is great for including images into program files. There are two versions, for X10 and X11. No compression. 1 bit color depth supported.

XPM - Xpixmap

The colored XBM equivalent.

XWD - X-Window Dump

This is the format of the Unix tool xwd, which snaps the contents of windows in X and saves them to a file. It was released with X10 and was extended with X11, for higher flexibility and true color capability. No compression.

XGA - X-Giga-Format

A format by Dieter Fiebelkorn for his snapshot tool. Saves data in 16 bits color depth and without header.

1G59131591