

PicCon

Morten Eriksen

COLLABORATORS

	<i>TITLE :</i> PicCon		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY	Morten Eriksen	August 15, 2024	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1	PicCon	1
1.1	PicCon Help	1
1.2	PicCon/Introduction	1
1.3	PicCon/SNES	2
1.4	PicCon/Sega Megadrive	2
1.5	PicCon/Emplant	3
1.6	PicCon/AHRM	3
1.7	PicCon/Requirements	3
1.8	PicCon/Installation	3
1.9	PicCon/FilesInDistribution	3
1.10	PicCon/Usage	4
1.11	PicCon/Usage/Scan modus	5
1.12	PicCon/Usage/Tutorials	6
1.13	PicCon/Usage/Tutorials	6
1.14	PicCon/Usage/Tutorials	6
1.15	PicCon/Usage/Tutorials	7
1.16	PicCon/Usage/General info	7
1.17	PicCon/Disclaimer and Author Info	7
1.18	PicCon/How To Register	8
1.19	PicCon/Project	8
1.20	PicCon/Project/Open picture	9
1.21	PicCon/Project/Open IFF	9
1.22	PicCon/Project/Load image	9
1.23	PicCon/Project/Load palette	10
1.24	PicCon/Project/Save image	10
1.25	PicCon/Project/Save palette	10
1.26	PicCon/Project/Save grid	10
1.27	PicCon/Project/Save data as...	10
1.28	PicCon/Project/Change screenmode	12
1.29	PicCon/Project/Modify palette	12

1.30 PicCon/Project/Load prefs	12
1.31 PicCon/Project/Save prefs	12
1.32 PicCon/Project/Info	12
1.33 PicCon/Project/About	12
1.34 PicCon/Project/Iconify	12
1.35 PicCon/Project/Quit	12
1.36 PicCon/Edit	13
1.37 PicCon/Edit/Cut frame	13
1.38 PicCon/Edit/Grab frame	13
1.39 PicCon/Edit/Box frame	13
1.40 PicCon/Edit/Set frame	14
1.41 PicCon/Edit/Free frame	14
1.42 PicCon/Edit/Autocrop	14
1.43 PicCon/Edit/Autoscan	14
1.44 PicCon/Edit/Flip	15
1.45 PicCon/Settings	15
1.46 PicCon/Settings/Miscellaneous	15
1.47 PicCon/Settings/Imageformat	15
1.48 PicCon/Settings/Imageformat/Bitplanes	16
1.49 PicCon/Settings/Imageformat/Interleaved	17
1.50 PicCon/Settings/Imageformat/Mask	17
1.51 PicCon/Settings/Imageformat/Chunky	18
1.52 PicCon/Settings/Imageformat/Sprite	19
1.53 PicCon/Settings/Imageformat/WBIcon	21
1.54 PicCon/Settings/Imageformat/Fontset	22
1.55 PicCon/Settings/Imageformat/IFF	23
1.56 PicCon/Settings/Imageformat/SNES modes	23
1.57 PicCon/Settings/Imageformat/SNES hardware	25
1.58 PicCon/Settings/Imageformat/Sega	28
1.59 PicCon/Settings/Imageformat/Megadrive hardware	29
1.60 PicCon/Settings/Paletteformat	30
1.61 PicCon/Settings/Paletteformat/4bits	31
1.62 PicCon/Settings/Paletteformat/8bits	31
1.63 PicCon/Settings/Paletteformat/32bits	31
1.64 PicCon/Settings/Paletteformat/LoadRGB4	31
1.65 PicCon/Settings/Paletteformat/LoadRGB32	31
1.66 PicCon/Settings/Paletteformat/IFF	32
1.67 PicCon/Settings/Paletteformat/SNES 5 bits	32
1.68 PicCon/Settings/Paletteformat/Megadrive 3 bits	32

1.69 PicCon/Settings/Paletteformat/Copperlist	32
1.70 PicCon/Settings/Grid settings	33
1.71 PicCon/Settings/Grabpen	34
1.72 PicCon/Misc	34
1.73 PicCon/Misc/Fake SNES screen	34
1.74 PicCon/Misc/Remake Amiga screen	34
1.75 PicCon/Misc/Pen usage in frame	34
1.76 PicCon/Misc/Remap pens in frame	35
1.77 PicCon/Misc/Trace unused pens	35
1.78 PicCon/Misc/Convert EHB->normal	35
1.79 PicCon/Misc/Reselect systempens	35
1.80 PicCon/History	35
1.81 PicCon/Thanks to...	38

Chapter 1

PicCon

1.1 PicCon Help

[Introduction](#)

[Requirements](#)

[Installation](#)

[Files in Distribution](#)

[Usage](#)

[Disclaimer and Author Info](#)

[How To Register](#)

[Thanks to...](#)

[History](#)

1.2 PicCon/Introduction

PicCon

Copyright © 1993 - 1994

All Rights Reserved

Written by

Morten Eriksen

PicCon is short for "Picture Converter". This is a utility made for programmers, which will convert IFF ILBMs (plus any pictureformat you've got support for in your datatypes library) to an appropriate image format. This is an essential stage mainly in the development of games, but is also useful in development of other software (like demos, applications, etc.). Not only whole pictures can be converted, but also parts of pictures can be cut out to be saved as e.g. sprites or small bitplanes.

Features include (those marked with (3.0 only) works only with OS3.0 (or later)):

- o Uses datatypes.library to read pictures (can load and process e.g. JPEGs and GIFs if you've got the datatypes) (3.0 only). If wanted, PicCon can also be run without the datatypes.library present, as custom IFF ILBM loading is included.
 - o Compatible with all screenmodes on OCS, ECS and AGA-chipsets.
 - o Display independent code and use of Commodore standard screenmoderequester (2.1+ only) secures that PicCon will run in any displaymode on a capable monitor.
-

- o Correct handling of all screenmodes, including extra halfbrite (EHB), Hold-And-Modify (both HAM6 and HAM8) and Dual Playfields (DPF).
- o Saving in numerous image formats, like ordinary bitplanes, chunkymode, sprites (even AGA-specific spriteformats), IFF ILBM, workbench icons plus more.
- o Saving of palettes in many formats, both ECS-compatible (4-bits-per gun, 12 bits per entry) and AGA-compatible (8-bits-per-gun, 24 bits per entry), OCS/ECS copperlists, AGA copperlists, IFF, and more.
- o Manipulation of cutted frame (autocropping and flipping).
- o Choose to save data as binary, assembly, C, E or Pascal source for including directly into your own sourcecode. Or save as linkmodules for inclusion in the linking process (handy for larger projects).
- o Automatic leftaligning of frames and blanking of trailing bits and bytes upon saveoperations (no need to keep your images on 8-pixel alignments).
- o Gridsave for saving loads of blocks of images in one go (for e.g. maptiles, frames in a spriteanimation, ...).
- o Special autoscan modus for fast processing of whole screens packed with graphical objects to be converted. It would even be possible to have all sprite- and animationframes your game contains in a single screen, and convert them all in one go (!).
- o Some often used imageprocessing functions included (some of which are very handy for programmers, though not included in any bitmap drawing-packages).
- o Extensive **SNES** support.
- o **Sega Megadrive** support.

NOTE:

PicCon is shareware software. If you find it useful, please register to get the full version. Please refer to the **How to Register** section of this documentation.

The difference between the demoversion and the registered version is that the demoversion has the **Save grid** and **Autoscan** functions disabled, and will exit after you've done 2 saves.

Please notice that you are not entitled to use the demoversion on a regular basis without registration. The demoversion is only for evaluation of the program. If you don't use it, delete it.

Your name and a personal code will be inserted in the registered version of PicCon.

1.3 PicCon/SNES

SNES is an abbreviate for Super Nintendo Entertainment System, also known as the Super Famicom in the U.S. of A.

SNES support in an Amiga graphicsconverter is not such a stupid idea as you might think for several reasons:

- 1) Lots of people do SNES development partly or fully on the Amiga.
- 2) The **Emplant** SNES emulation will probably be ready soon (this was written in March '94).
- 3) Parts of the Amiga and SNES demoscenes (yes, there really is one!) 'overlap', by which I mean to say that many Amigacoders also program for the SNES.

The SNES support in PicCon was implemented by request from Daniel Hansson. Thanks for your help, mate!

1.4 PicCon/Sega Megadrive

The Sega Megadrive is also known as the Genesis in the U.S. of A.

Sega Megadrive support in an Amiga graphicsconverter is not such a stupid idea as you might think for several reasons:

- 1) Lots of people do Megadrive development partly or fully on the Amiga.
 - 2) The **Emplant** Megadrive emulation will probably be ready soon (this was written in April '94).
 - 3) Conversions are often done between Amiga and Megadrive software (the main CPU of the Megadrive is a MC68000).
-

1.5 PicCon/Emplant

Emplant is a plug-in card developed by Utilities Unlimited that make it possible to flawlessly emulate just about anything in the homecomputermarket on your Amiga. As I'm writing this (March '94), the Mac emulation has been on the market for several months, the IBM-PC compatibles (i486) emulation is due real soon now and lots of other goodies is supposed to follow (like C64, SNES and MegaDrive/Genesis emulation).

1.6 PicCon/AHRM

AHRM is an abbreviation for the Amiga Hardware Reference Manual, which should be present in any good Amiga coder's personal library!

(The ISBN code for the second printing is 0-201-18157-6).

1.7 PicCon/Requirements

REQUIREMENTS

Any Amiga with Kickstart 2.04 (or greater).

Tested on:

- * A500, A1200, A4000
- * 68000, 68EC020, 68EC030, 68040
- * Kickstarts 37.175, 39.106
- * 0.5 - 2.0MB CHIPRAM, 0 - 6MB FASTRAM
- * All screenmodes on all monitortypes distributed with OS3.0 (except the A2024 monitortype, which I haven't been able to check up)

Also tested with Enforcer and Mungwall. As far as I can tell, PicCon is free from hits (please let me know if you find any).

If you find any bugs under your configuration, please contact me (address in the [Disclaimer and Author Info](#) section).

1.8 PicCon/Installation

INSTALLATION

Just drag the PicCon drawer to the place on your harddisk where you want to install PicCon.

Also make sure that if you want to take advantage of the datatypes.library, you have to have datatypes.library installed in LIBS:, some picture datatypes installed in SYS:Classes/DataTypes and the corresponding datatypes descriptors in Sys:Devs/DataTypes. (Take a look at the WorkBench disks supplied with your Amiga to get this set up correctly).

1.9 PicCon/FilesInDistribution

FILES IN DISTRIBUTION

PicCon/PicCon - main program

PicCon/PicCon.info - workbench icon for program

PicCon/PicCon.guide - AmigaGuide Help/Documentation for PicCon

PicCon/PicCon.guide.info - workbench icon for docsfile

PicCon.info - drawer icon

PicCon/Extra/

PicCon/PicCon/

PicCon/MagicWB/

In the PicCon/Extra/ drawer:

Example1 - example image file #1 (for the quick tutorials)

Example2 - example image file #2 (for the quick tutorials)

In the PicCon/Sources/ drawer:

5planespic.s - asmsource for displaying a picture in 5 planes (PAL)

8planespic.s - asmsource for displaying a picture in 8 planes (PAL)

6planesHAMpic.s - asmsource for displaying a HAM6 picture (PAL)

8planesHAMpic.s - asmsource for displaying a HAM8 picture (PAL)

16pixSprite.s - asmsource for a moving 16 pixels wide sprite

64pixSprite.s - asmsource for 4 moving 64 pixels wide sprites

init.i - initialize and reset system routines

includes.i - system includes

sprite16.raw - a spriteimage to go with the 16pixSprite.s source

sprite64.raw - spriteimages for the 64pixSprite.s source

(Note about the sources: all sources have been compiled successfully with AsmOne, DevPac and PhxAss. The sources are only meant as a help for novice assembler coders on the Amiga. Be aware that all sources shut down the multitasking, and if you have a problem with this, don't use them. The two *.raw files only serve as examples to use in the spritesources (you have to make the other raw-files yourself (with PicCon), as they were too big to include in the archive)).

In the PicCon/MagicWB/ drawer:

PicCon.info - a very nice icon to use for PicCon if you've installed MagicWB. Thanks to Niels Jørgensen for the sending me the icon!

1.10 PicCon/Usage

USAGE

PicCon can either be started from CLI by typing its name in a shell/cli-window, or by doubleclicking it's icon.

Upon CLI-startup, PicCon can be run in it's special **Scan modus**.

On startup you'll be presented with information on the version of the program and the license (or information on how to register if you are using the demoverison).

After clicking "Ok" you'll have the program waiting for input through the menubar or the keyboard. Click below for full explanation on each of the menuchoices:

Project Edit Settings Misc

..or here for some other information concerning the use of PicCon:

General info

I've also whipped up a few quick tutorials for some of the functions:

Grab frame

Autoscan

Remap pens in frame

Trace unused pens

1.11 PicCon/Usage/Scan modus

If PicCon is started from the command line with exactly 3 parameters, it'll enter it's special scan modus. The effect of this is the same as if you (from inside PicCon) load a picture and then use **Autoscan**. (If PicCon is started with any other number of parameters than 3, it will start up in it's normal way).

The syntax for the 3 parameters is like this:

PicCon [picturefile] [savefile] [boxpen#]

PicCon will then start, load the picturefile, scan the picture and save all the frames boxed in the boxpen, and join them to the savefile. The settings that decide the imageformat and the saveformat will be taken from the S:PicCon.prefs file (as the number of different possibilities you have with the various imageformats is getting pretty high, this was the best way of doing this that I could think of).

This option should lead to considerable savings of time spent mucking about in graphics converters during big projects (like games), as this option makes it easy (with a bit of thought) to save all the graphical images your project contains with a single command in the CLI!

Now, how is this possible? Easy, just use this approach:

1. Make boxes in a specified pen on the picture which your graphician should draw the images (used in the project) inside. Keep only images of the same type on the same pictures (do not mix f.ex. spriteimages and interleaved blitterdata).
2. Make a drawer on your disk (f.ex. MyDisk:ScriptFiles), into which you put the preferencesfile for each imagetype you're using in your projects.
3. Make a scriptfile that in turn copies the prefs-files to S: and runs PicCon.

Here's a quick example just to make things completely clear:

Let's say I'm making a game. In this game I use a whole bunch of images in spriteformat, 16 pixels wide attached and with no CTRL words, (for animated parts of a mothership, or something...) and I need a lot of images in interleaved format, all with an extra blitterword attached to the left of the frames (for the rest of the moving objects on the screens (BOBs)). All sprites have to use the same palette, and all BOBs also use the same palette. Before I start this project, I make two huge screens in DPaint with loads of boxes (in the same color), maybe with some descriptive text in it, to show my graphician what I need and how I want the graphics. After the graphician now has drawn some (or all) of the frames I need, I enter PicCon, set the imageformat to "Sprite 16, Attached on, CTRL words off" and set the saveformat to "Binary". All the images are of the same size, the .offsets of the individual frames are not interesting, so I also turn off Save Autoscan offsets in the **Miscellaneous** menuoption. I then save the preferences file (with **Save prefs**). I make a directory on my harddisk (MyDisk:ScriptFiles) and copy the S:PicCon.prefs file to this drawer as "Prefs1". I now set the imageformat to "Interleaved, Blitterwords left" and save the prefs again. I copy the S:Piccon.prefs file to MyDisk:ScriptFiles/Prefs2. Then I make a script that looks like this (assume the boxes have been drawn with pen #15):

```
copy S:PicCon.prefs MyDisk:ScriptFiles/TempPrefs
copy MyDisk:ScriptFiles/Prefs1 S:PicCon.prefs
PicCon MyDisk:IFFs/SpriteFrames MyDisk:RAWs/SpriteFrames 15
copy MyDisk:ScriptFiles/Prefs2 S:PicCon.prefs
PicCon MyDisk:IFFs/BlitFrames MyDisk:RAWs/BlitFrames 15
copy MyDisk:ScriptFiles/TempPrefs S:PicCon.prefs
delete MyDisk:ScriptFiles/TempPrefs
```

Then just set the 'script'-bit in the scriptfile's modebits, and each time your graphician has changed some of the graphics, and you want to try out how it looks in your game, just enter the name of the scriptfile in the CLI, and seconds after you've got the files containing the raw, ready-to-use images in your 'MyDisk:RAWs' drawer. Not bad, eh? ;)

BTW, to avoid having requesters pop up to inform you about what you saved, etc, just remove the requesters (in the **Miscellaneous** settingswindow) before you save the preferencesfiles. If you do this, no user interaction will be necessary at all (unless something goes wrong).

If anything in this explanation seemed unclear, please check out the documentation on the **Autoscan** function (if you're still stuck, try playing around a bit, look at the size of the savefiles, etc).

1.12 PicCon/Usage/Tutorials

Quick tutorial to show the use of the AutoScan function:

1. Load the file "Example2".
2. Hit 'z' or select **Autoscan**.
3. Choose the filename for the savefile.
4. Pick the correct boxpen from the palette (which is the last-but-one entry (pen #14)).
5. The 5 frames in the boxes will now be saved to the file you specified, and the offsets into the file for the individual frames will be saved to the file "[filename].offset". The frames will be saved in the format you've specified in **Imageformat**.

This way, it's possible to save all the frames in e.g. a game in one go (!). This should lead to considerable savings of valuable time for most programmers.

1.13 PicCon/Usage/Tutorials

Quick tutorial to show the use of the Grab function:

1. Load the file "Example2".
2. Hit 'g' or select **Grab frame**.
3. Pick the correct boxpen from the palette (which is the last-but-one entry (pen #14)).
4. Click somewhere inside one of the boxes. The inner frame will now be selected, just as if you'd framed it manually.
5. Do your stuff here (like saving, flipping, cropping etc).
6. Repeat the process by again hitting 'g'. Notice that you'll not be requested for the boxpen any more, as it's assumed it's the same as the first time you 'grabbed' a frame. To change the pen by which you wish to grab boxed frames, choose **Set new grabpen**.

1.14 PicCon/Usage/Tutorials

Quick tutorial to show the use of the Remap function:

1. Load the file "Example1".
2. Notice that the graphician here did two mistakes;
 - a) he filled the background with pen #11 instead of pen #0 (which should always be used for the background if you're doing graphics for e.g. games)
 - b) some of the (dark) pixels inside the bee-image is drawn with pen #0, which is not good, as any background graphics will show through it if it's moved and animated over other graphics (as it would be in a game)

Now, how should this be solved? Using DPaint, you'd have to either redraw parts of the image by hand (ugh!), or you'd make a stencil (which sounds easy, but in most cases it's very cumbersome). Instead you should use this function in PicCon.

3. Hit 's' to get the **Set frame** window. Now change the x2-value to 71, and y2 to 54. The brush has now been framed.
4. Select **Remap pens in frame** from the **Misc** menu. Pick pen #0, hit "Swap", hit pen #11, hit "Done".
5. To get the colorvalues correct, now choose **Modify palette** from the **Project** menu. Pick pen #0, hit "Swap", hit pen #11, hit "Ok".

Voila!

6. After this, you might want to select **Reselect systempens** to get menus and requesters that would be nicer to look at.
-

1.15 PicCon/Usage/Tutorials

Quick tutorial to show the use of the Compress pens function:

1. Load the file "Example1".
2. Hit 'i' to bring up the Info-box. Notice that the picture uses 5 bitplanes (32 colors).
3. Choose **Trace unused pens** from the **Misc** menu.
4. You'll be prompted with a requester telling you that 20 colors are unused, and if you want the palette to be compressed. Select 'Yes' to shift all used pens to the lowest pennumbers.
5. You'll now be prompted with a new requester, asking you if you want to remove the extra bitplane. Answer 'Yes' again.
6. The picture is now reduced to 4 bitplanes (hit 'i' to conform this), and the unused pens is the 4 equal colorvalued pens at the back (as you can see by choosing **Modify palette** from the **Project** menu).

1.16 PicCon/Usage/General info

Here are some general hints about using PicCon:

- o You can modify the dimensions of the current frame by using the cursorkeys. Cursorkeys only to expand frame, shift + cursorkeys to reduce size of frame.
- o The option to swap the picturecolors with the WB colors has been removed as this option should in most cases be superfluous in version 2 (and later) of PicCon, as the program is now coded to make controlled use of the systempens. Only the most extremely strange palettes (like all monocolored entries) should cause the menus and requesters to be rendered unreadable. Note that not all systempens can be controlled under OS2.0, and therefore PicCon will switch in 4 standard paletteentries when you use menus in PicCon on a 2.0 Amiga.
- o As you can see, the most used functions in the program have hotkeys. You can not only activate them by holding the Right Amiga key and pressing the appropriate hotkey, but also by pressing the hotkey alone. This way it's possible to operate PicCon by keeping one hand on the keys, and one hand on the mouse (like e.g. DeLuxePaint). (Oh, BTW, try to find the "hidden mode"...)
- o The cutted box' dimensions, the current imageformat and the current paletteformat will usually be displayed in the screen titlebar. The image- and paletteformats are shortened (so I could be sure that it always fits in the smallest screens). The shortversions shouldn't be so hard to figure out, e.g. Imageformat set to packed chunky and **Save data as** set to assemblersource would insert this on the titlebar: CHNK/PCK/ASM, and a spritesave with attachment and CTRLwords saved as a linkobject: SPR32/AC/LNK.
- o The RAW files which are saved from PicCon have some information regarding their format and dimensions set in the file's Comment-field. Use the DOS command 'list [filename]' to see what I mean. This format is pretty easy to understand too: first there's a 4 letter abbreviate of the format (e.g. BPLS for bitplanes), then there's Width x Height x Depth, and finally there's a longword that describes the screenmode ID used in PicCon upon saving. This feature can be turned off in the **Miscellaneous** settings window.

1.17 PicCon/Disclaimer and Author Info

PicCon Information

PicCon Copyright © 1993 - 1994 Morten Eriksen, All Rights Reserved.

The registered version is copyrighted, and is a personal license only. The demoversion is freely distributable as long as all of the files are included in their original form without additions, deletions or modifications of any kind, and only a nominal fee can be charged for the distribution. This software is provided "AS IS" without warranty of any kind, either expressed or implied. By using PicCon, you agree to accept the entire risk as to the quality and performance of the program.

Please send your comments, wishes and bug reports for PicCon to:

Morten Eriksen

Jonsborgv. 18

7563 Malvik

NORWAY

Phone: +47-73977371

Or by email (I love getting email!):

mortene@stud.unit.no

Remember to state the version number you are using, in all correspondance.

1.18 PicCon/How To Register

How To Register

To register, send me an errorfree double-density disk and USD \$15 (or GBP £10, FFR 75, DM 25, NKR 100, SKR 100, DKR 100, or an equal amount in any currency). Cash only, please. This will earn you the latest full version plus any number of free upgrades.

If you've got the possibility of receiving the package through email, send only money (no disk), and let me know how to contact you.

For future upgrades for registered users; send money to cover for a disk, packing and postage (one 2DD disk is approx. \$1 USD, packing approx. \$1 USD and postage is approx. \$1.5 inside Europe and approx. \$2 USD outside, which should sum up to ~\$3.5 - \$4 USD). This is not necessary if you've got an email adress where you can receive the upgrade, as they will be automatically mailed to you.

Notice that the demoversion is not freeware. If you use it regularly, you should register.

Mail to:

Morten Eriksen

Jonsborgv. 18

7563 Malvik

NORWAY

Phone: +47-73977371

Email (send me lots of email!):

mortene@stud.unit.no

Also contact me for questions, ideas for future releases, bugreports, wanted new formats (no matter how weird! Anyone want to have 3DO support? C64 sprites? Sure! Just inform me on the format specifications/limits), or anything else. Most of the new features and other program improvements from version 1 to 2 were suggestions and demands from the users.

Please support Amiga shareware, don't spread the registered version, nor use a pirated version. Increased support means better and more software for the Amiga range of computers.

1.19 PicCon/Project

Menuchoices on the "Project"-menu:

Open picture

Open IFF

Load image

Load palette

Save image
Save palette
Save grid
Save data as
Change screenmode
Modify palette
Load prefs
Save prefs
Info
About PicCon
Iconify
Quit

1.20 PicCon/Project/Open picture

Choose the picturefile to load. PicCon will load a picture in any picture format which you have support for in your datatypes library. The IFF ILBM datatype specification is delivered as standard with the distribution of WorkBench3.0, but there's many more in the Public Domain and as shareware (including JPEG, GIF, PCX and BMP).

After the file has been loaded and unpacked (this might take some time, especially with JPEG-pictures), you'll be presented with a requester displaying some information on the picture. Click "Ok", and the picture will now be displayed on your screen (the "Ok"/"Cancel" requester can be turned off in the **Miscellaneous** settings window if it annoys you). From here on you can start cutting out and saving pieces of the picture in many different image formats, save this picture's palette, etc.

Loading through the datatypes library may only be done on Amigas with OS3.0 (or later). On 2.0 Amigas, you have to use **Open IFF** to load pictures.

1.21 PicCon/Project/Open IFF

This option exists in addition to **Open picture** for those who only have OS2.0 (or for any other reason don't want to use the datatypes.library).

1.22 PicCon/Project/Load image

This function will load RAW data and display it as if it were loaded as an ordinary picture. Useful for loading old RAW files (for which you've deleted the IFF ILBM-file), among other things.

When selecting this function, PicCon assumes that you want to load a RAW file in the same format as you have specified in the **Imageformat** settings. With some formats you must also specify width, depth and displaytype (a requester will pop up, allowing you to set these parameters). After setting these options, you're requested to select the screenmode you want the file to be displayed in. When the frame has been loaded, you're free to treat it as any file loaded with **Open picture** or **Open IFF** (cut frames, save images etc).

With this function, you can even load the images contained in workbench icons and Amiga fontsets (all **imageformats** are supported, except SNES and Megadrive RAW character set).

Note that this option ignores the **Save data as** setting, as only binaries can be loaded.

See also **Save image**.

1.23 PicCon/Project/Load palette

This function loads and inserts a palette into the picture currently displayed. The paletteformat is assumed to be the same as the one you've specified in the **Paletteformat** settings window.

Note that this option ignores the **Save data as** setting, as only binaries can be loaded.

See also **Save palette**.

1.24 PicCon/Project/Save image

Save the picture information contained in the frame in your chosen image format. If no frame is cut out, this option will save the whole picture in your specified format. See also: **Imageformat** and **Save data as**. The dimensions of the selected frame is normally displayed in the menubar.

1.25 PicCon/Project/Save palette

Save this picture's palette in your specified format. See also: **Paletteformat** and **Save data as**.

1.26 PicCon/Project/Save grid

Save a bunch of frames ordered in a grid. Very useful for saving maptiles, groups of spriteanimationframes etc. This function will save the tiles you've decided to save (with the **Grid settings**) exactly in the same way as if you've saved all tiles by hand, and then joined together the files.

See also: **Grid settings** and **Save data as**.

1.27 PicCon/Project/Save data as...

Choose if the saved data should be output as RAW binary, assemblysource, C source, Pascal source or as a linkable objectmodule.

Data output as assembly sourcecode will look like this:

```
SECTION mydata,DATA
```

```
dc.b $45,$fe,$12,$56,$12,$56,$12,$56,$45,$fe
```

```
dc.b $14,$2e,$32,$52,$1f,$fe,$ee,$16,$23,$23
```

```
.  
.
.
```

```
dc.b $a2,$ba,$11,$01
```

C sourcecode will be saved in this format:

```
UBYTE myData[] =
```

```
{
```

```
0x45,0xfe,0x12,0x56,0x12,0x56,0x12,0x56,0x45,0xfe,
```

```
0x14,0x2e,0x32,0x52,0x1f,0xfe,0xee,0x16,0x23,0x23,
```

```
.
```

```
.
.
0xa2,0xba,0x11,0x01
};
...while Pascal sourcecode will be saved in this format:
```

```
Const
myData : Array[$0..$[# of entries - 1]] of Byte =
(
$45,$fe,$12,$56,$12,$56,$12,$56,$45,$fe,
$14,$2e,$32,$52,$1f,$fe,$ee,$16,$23,$23,
.
.
.
$a2,$ba,$11,$01
);
```

...and E source:

```
myData:
CHAR $45,$fe,$12,$56,$12,$56,$12,$56,$45,$fe,
CHAR $14,$2e,$32,$52,$1f,$fe,$ee,$16,$23,$23,
.
.
.
CHAR $a2,$ba,$11,$01
```

These examples show how a sourcecode will be saved in byte-format, but PicCon recognizes if word- or longwordformat would be more appropriate. E.g.: 16 pixels wide sprites will be saved with two words on each line, while 64 pixels wide sprites will be saved with 4 longwords on each line.

NOTE:

Often you want the saved data to reside in CHIP RAM. To obtain this, you'll have to specify for your compiler to place it in a chipdata section. In assembly you'll do this by appending either 'CHIP' or '_C' to the 'DATA' keyword, e.g.:

```
SECTION mydata,DATA_C
```

All assemblycompilers may not be completely compatibel with this use of the CHIPMEM option, check your manual.

In SAS/C, you'll insert the keyword 'chip' between the 'UBYTE' keyword and the name of your datasection (e.g. myData):

```
UBYTE chip myData[] =
{
.
.
.
};
```

When saving as a linkmodule, you'll be asked to specify the external definition (which is the name of the pointer that points to the beginning of the data. This pointer can later be referenced in your sourcecode with "xref [name]" (assembler) or e.g. "extern void *[name]" (C-source)). You must also specify the type of memory you want the module to be loaded to.

1.28 PicCon/Project/Change screenmode

Sometimes when you load a picture it will be loaded in a screenmode which is not very convenient for cutting out pieces of the picture (pixelperfect framing in SuperHires is not very easy...). Then use this menuchoice to change the screenmode. All screenmodes contained in your Monitor-drawer (in DEVS:Monitors/) will be listed out for you to choose from.

The screenmode requester wasn't available in the Amiga's operating system until version 2.1, so this option will be ghosted if you're running on 2.04 or 2.05.

1.29 PicCon/Project/Modify palette

A standard paletterequester has now been included in PicCon for editing the paletteentries on the fly. If you've ever used a paletterequester before, no explanations should be needed on this option. :)

1.30 PicCon/Project/Load prefs

Load and set the preferences in the S:PicCon.prefs file. This is also automatically done upon starting PicCon. See [Save prefs](#) too.

1.31 PicCon/Project/Save prefs

Save settings to S:PicCon.prefs. This file will be automatically loaded upon startup, but can also be done manually by choosing [Load prefs](#) from the menu.

Settings included in the preferences file are: savetype (binary/ assemblysource/C source/linkmodule), image format, palette format, last used directory and filenames on loading/saving, plus lots of other settings and some internal data.

This option is handy if you grow tired if reselecting the same directory/ filename/menuoptions every time you load PicCon.

1.32 PicCon/Project/Info

Displays information about the picture and the framed box.

1.33 PicCon/Project/About

Displays information about the program, the author and the license.

1.34 PicCon/Project/Iconify

Frees up the memory taken by the screen and the window that PicCon uses. Select the iconify-bar that pops up on the Workbench and click the right mouse button to reenter the program.

1.35 PicCon/Project/Quit

Exits program.

1.36 PicCon/Edit

Menuchoices on the "Edit"-menu:

Cut frame
Grab frame
Box frame
Set frame
Free frame
Autocrop
Autoscan
Flip X
Flip Y

1.37 PicCon/Edit/Cut frame

After selection of this menuitem, your mousepointer will be followed by a cross. Upon pressing the left mousebutton, you now have the possibility of selecting an appropriate piece of your picture for saving in RAW format. Just keep the left mousebutton pressed while you drag the mousepointer so that you boxes in the piece you want to save. See also [Grab frame](#), [Box frame](#) and [Set frame](#).

Notice that your mousepointer will contain coordinates which tell you the current position in your picture, and when framing, the current size of your framebox. Upon 8-pixels boundaries, the coordinates in the mousepointer will change color to indicate this. Notice also that you don't need to put your frames on 8-pixel offsets and bytemultiple boundaries to get a correct saved image. PicCon will automatically adjust, left-align and blank any trailing bits and bytes on the boxed frames upon any saveoperations.

If your picture is 'open' (clearly seperated items on the screen) you can use the [Autocrop](#) option after selection to avoid having to do pixelperfect boxing of your frame.

If you misplace the selection, you might want to use the [Set frame](#) option to correct some of the coordinates.

1.38 PicCon/Edit/Grab frame

If you keep the graphical objects you want to cut out in boxes (f.ex. to guide your graphician, so he doesn't draw too big objects), then you can use this option to frame the images. Just select the pen by which the box is drawn, and click somewhere inside the box.

Notice that the box have to be complete, "holes" in the boxwalls will make the box unusable.

If you've got many frames stored in this "boxed" format, you might consider using the [Autoscan](#) function, which will save all your boxes in one go.

The second time you use this option, you will not be requested for the boxpen, as it's assumed that it is the same as the first time you use this function. Change the selected boxpen with the [Set new grabpen](#) setting. Please note that there is no restrictions on the boxpen concerning usage in your frames (the algorithm will only react on complete boxes).

See also [Cut frame](#), [Box frame](#) and [Set frame](#) plus the quick tutorial: [Grab frame](#).

1.39 PicCon/Edit/Box frame

This function let you cut out your frame with a fixed sized box. The size of the box is set from the current frame, or manually in case there is no cutted frame.

This is handy if you want to cut several pieces with the same size out of your picture. See also [Cut frame](#), [Grab frame](#) and [Set frame](#).

1.40 PicCon/Edit/Set frame

Instead of selecting a frame by dragging out the framebox, you can use this function to manually input the coordinates of the corners of the framebox. See also [Cut frame](#), [Grab frame](#) and [Box frame](#).

This function is also handy if you miss slightly when you use the mouse to select the framebox; just pop up the coordinates inputbox and modify the coordinates.

1.41 PicCon/Edit/Free frame

Frees the selected framebounds (by setting the max dimensions).

1.42 PicCon/Edit/Autocrop

After framing a piece of the picture, you might want to use this option to make the framing 'perfect' (no extra space included). This function excludes the need to do pixelperfect framing.

1.43 PicCon/Edit/Autoscan

If you're working on large projects, it's very timeconsuming to manually cut and save the individual frames. With a bit of planning and the help of this option, you can greatly reduce the time spent in PicCon. First you have to keep all the graphical objects in boxes, as you would with the [Grab frame](#) function (and all boxes have to be in the same color). Second, select your wanted imageformat (e.g. Bitplanes, Sprites or whatever). Now use this option to get PicCon to automatically scan and save all boxed frames in one go. The frames will be saved to a single file, while the offsets to the different frames in the file will be saved to another file with the same name, but with ".offset" appended to the filename.

The frames will be saved top-to-bottom and left-to-right, which means that frames with lower topmost y coordinates will be appended first to your savefile, while frames with identic topmost y coordinates will be saved according to leftmost x coordinates (lowest leftmost x coordinate -> saved first).

Note that it is the inner box that will be saved (the borders are not included). So to save e.g. a 64x64 frame, you'd have to make a 66x66 pixels wide box (including the borders). There is no restrictions on the use of the boxpen otherwise in your picture, other than that all "insides" of boxes made with that particular pen will be saved to the file (single pixels drawn with the boxpen in your frames will be discovered and not reacted on in the autoscan-algorithm, so your graphician is free to use all the available pens in his masterpieces...).

Note also that this function will work exactly as if you'd cutted and saved all the frames individually, and then joined the files afterwards.

The .offset-file will consist of ASCII-text, in this format:

```
frame_000 EQU $00000000
```

```
frame_001 EQU $00004000
```

```
frame_002 EQU $00007800
```

```
.
```

```
.
```

```
frame_xxx EQU $yyyyyyyy
```

(Labels on the left side, offsets on the right side. This format should be compatible with most assemblers).

See also the quick tutorial: [Autoscan](#).

1.44 PicCon/Edit/Flip

Will flip your selected frame in the chosen direction (if no frame is cutted, the whole picture will be flipped).

This option will be useful if you don't want to have flip-routines in your code, but rather include the flipped frames as binary data.

1.45 PicCon/Settings

Imageformat

Paletteformat

Grid settings

Miscellaneous

Set new grabpen

1.46 PicCon/Settings/Miscellaneous

The options in the "Miscellaneous settings" window are supposed to be an easy way to configure PicCon so that you can remove things that might annoy you.

Set filecomments:

If this option is on, the RAW image files that PicCon saves will have information about the format and the dimensions of the frame included in the file's commentfield. Use the DOS-command 'list' on a file saved from PicCon to see what I mean.

Confirm loads:

Use this option to turn on or off the requester that pops up before the current picture is swapped with the one you want to load (the requester that gives you some information about the picture, and let you have the choice of cancelling a load).

Confirm overwrites:

Ask the user before any files get overwritten.

Inform saves:

Pop up a requester after you've saved a file to tell you what you actually saved.

Autosave prefs on exit:

If you always want to restart in the state you left PicCon.

Include pen 0 in penstats:

Often the frames contains mostly transparency, and in that case, the **Pen usage in frame** isn't very helpful. Then use this option to not include pen 0 in the penstatistics window.

Save Autoscan offsets:

If you use the **Autoscan** function, and all the frames in your picture are of the same size, you probably don't need the information contained in the .offsets-file.

1.47 PicCon/Settings/Imageformat

The settings in this window is for selecting the way the boxed frame will be saved when choosing **Save image** in the **Project** menu.

Settings in the "Imageformat"-window:

Bitplanes

Interleaved

Mask

Chunky

Sprites

WB icon

Fontset

IFF ILBM

SNES modes

Megadrive

1.48 PicCon/Settings/Imageformat/Bitplanes

Set RAW save to ordinary bitplaneformat:

<- framewidth ->

+-----+

|| ^

|||

| Bitplane 0 | frameheight

|||

|||

+-----+ v

|| ^

|||

| Bitplane 1 | frameheight

|||

|||

+-----+ v

|| ^

|||

| Bitplane 2 | frameheight

|||

|||

+-----+ v

|

|

.

.

etc

In conjunction with this format, you must also specify the "Blitterwords" option. If you set this format to any other than "None", an extra column of words (16 pixels, 2 bytes) will be saved to either the left or the right (or both) of the raw bitplane frame. This is necessary to be able to use the blitterchip's shiftmodus.

(Check your copy of the [AHRM](#)).

About HAM8 pictures: the two controlplanes in HAM8 pictures are supposed to be at the front (bitplane pointers in \$dff0e0 and \$dff0e4), but when setting up HAM8 screens through OS functions, they are inserted at the back of the eight planes. PicCon will save RAW HAM8 images with the controlplanes at the front.

1.49 PicCon/Settings/Imageformat/Interleaved

Set RAW save to interleaved format (useful for blittergraphics and fast displays):

```
+-----+-----+
|
|
|
|
| Bitplane 0 | Bitplane 1 |-- . . etc
|
|
|
|
|
+-----+-----+
```

In conjunction with this format, you must also specify the "Blitterwords" option. If you set this format to any other than "None", an extra column of words (16 pixels, 2 bytes) will be saved to either the left or the right (or both) of each of the interleaved raw plane frames. This is necessary to be able to use the blitterchip's shiftmodus.

(Check your copy of the [AHRM](#)).

About HAM8 pictures: the two controlplanes in HAM8 pictures are supposed to be at the front (bitplane pointers in \$dff0e0 and \$dff0e4), but when setting up HAM8 screens through OS functions, they are inserted at the back of the eight planes. PicCon will save RAW HAM8 images with the controlplanes at the front.

1.50 PicCon/Settings/Imageformat/Mask

If this option is selected upon saving, a mask of the current frame will be created and saved. The mask is created by OR'ing all the bitplane frames together, effectively making a 'shadow' of the object within the frame. Such frames are necessary to use when you want to animate or move blitter objects on top of some background graphics. Otherwise some of the background bitplanes will "crash" with your object's bitplanes. Just set up the blitter (or use the CPU) to perform this logic function on all common background- and objectplanes:

bckgnd mask result

```
0 0 0
0 1 0
1 0 1
1 1 0
```

(result = ~mask & bckgnd)

This is referred to as the cookie-cut method in the **AHRM**.

Along with this option, you must also choose if you want to save the frame with blitterwords (see **Bitplanes** or **Interleaved**), and if you want to save one maskframe, or if you want to save multiple interleaved frames:

If the "Single plane" option is set, the selected frame will be saved as a blittermask in this format:

```
framewidth
+-----+
^||
|||
|||
|||
||
frame- | Mask |
height ||
||
|||
|||
|||
v +-----+
```

...where the bits in "Mask" is made up by OR'ing together the individual bit- planes in the frame.

If the "Interleaved" option is set, the selected frame will be saved as a blittermask in this format:

```
+-----+-----+ +-----+
|||||
|||||
|||||
|||||
| Mask | Mask |.....| Mask |
| plane0 | plane1 || planeN |
|||||
|||||
|||||
+-----+-----+ +-----+
```

The mask will be saved as many times as there's bitplanes in the picture (N).

This format makes it possible to mask out unwanted bits in all bitplanes with only one blitteroperation (if you've made your screen interleaved).

1.51 PicCon/Settings/Imageformat/Chunky

Chunkymode is an often used IBM-PC format, but will also be used in the Amigascene with the introduction of the Akiko chip in the CD32-console (and besides; the IBM-PC **Emplant** emulation is supposed to be out soon).

In chunkymode, 1 byte is saved for each pixel in the picture. E.g.: If you've got 5 bitplanes, with bits set in planes 0 and 3, and unset in planes 1, 2 and 4, the pixel will have this value:

% 0 0 0 0 1 0 0 1 = \$09 = dec 9

bit# 7 6 5 4 3 2 1 0 (bits in unused bitplanes will be unset)

Along with the "Chunky"-option, you must also specify the "Packed"-setting. If you don't select packed chunky, the frame will be saved in the format explained above. If you do set the "Packed"-option, the pixels from each byte will be leftshifted to remove any superfluous bits (e.g. in the above example we'll get the value: %01001xyz, where xyz will be shifted in from the next byte in the framedata).

1.52 PicCon/Settings/Imageformat/Sprite

Will set RAW saveformat to Amiga sprites.

You must also set the spritewidth, attachment and CTRLwords options:

Spritewidth is set to either 16, 32 or 64 pixels. 32 and 64 pixels wide sprites might only be used with the AGA chipset. Spriteheight is unlimited on any version of the Amiga chipsets.

For displaying sprites with more than 3 colors, you'll have to make them attached. This means that you must overlap two sprites to get 15 colors to choose from. The sprites will be saved like this:

+-----+

||

||

| Sprite 0 |

||

||

+-----+

||

||

| Sprite 1 |

||

||

+-----+

(Check your copy of the [AHRM](#)).

Setting the "CTRLwords" option will make the program append some blank bytes in front of and behind the sprite(s) when saved. The bytes in front of the sprite is used to set position, attachment and other control bits. The bytes behind the actual imagedata is used for spritetermination.

The number of bytes appended to the actual spritedata will depend upon your selected spritewidth. If you're using 16 pixels wide sprites, 2 words will be appended in front of the sprite(s) and 2 words behind. For 32 pixels wide sprites, you'll get 2 longwords extra in front and behind, and for 64 pixels wide sprites it'll be 2 double longwords.

(Check your copy of the [AHRM](#)).

Notice that if your frame is wider than the width of one sprite (16, 32 or 64 pixels), several sprites will be appended to the same savefile. This makes it possible to save a whole row of sprites in one operation. E.g. a 48-pixels wide frame of 16-pixels wide, attached sprites with CTRLwords will be saved like this:

+-----+

| CTRLwords |

+-----+ ^
| | |
| sprite #0 |
| | spriteheight
| planes 0 + 1 |
| | |
+-----+ v
| termination |
+-----+
+-----+
| CTRLwords |
+-----+ ^
| | |
| sprite #0 |
| | spriteheight
| planes 2 + 3 |
| | |
+-----+ v
| termination |
+-----+
| CTRLwords |
+-----+ ^
| | |
| sprite #1 |
| | spriteheight
| planes 0 + 1 |
| | |
+-----+ v
| termination |
+-----+
+-----+
| CTRLwords |
+-----+ ^
| | |
| sprite #1 |
| | spriteheight
| planes 2 + 3 |
| | |
+-----+ v

```

| termination |
+-----+
| CTRLwords |
+-----+ ^
| | |
| sprite #2 |
| | spriteheight
| planes 0 + 1 |
| | |
+-----+ v
| termination |
+-----+
+-----+
| CTRLwords |
+-----+ ^
| | |
| sprite #2 |
| | spriteheight
| planes 2 + 3 |
| | |
+-----+ v
| termination |
+-----+

```

1.53 PicCon/Settings/Imageformat/WBIcon

This option sets the image format of the savefile to standard Amiga workbench icon. The format and type of the icon must also be set:

Formats supported:

WBDISK:

Same type as e.g. the RAM Disk icon.

WBDRAWER:

Same type as any drawer in your workbench.

WBTOOLS:

Executable files (and scriptfiles).

WBPROJECT:

Files that need a program to "treat them", e.g. docfiles.

WBGARBAGE:

Same type as the standard Amiga garbage can.

WBDEVICE:

I don't know what kind of files that use this icontype, as I don't think I have ever seen a "device-icon". I can't save icons of this format either, as they have no default in the 3.0 workbench (well, not in my workbench, that is). Maybe this is a future format?

WBKICK:

These icons can be saved under my configuration, but I still don't know what they are good for, as I have never seen any files with a WBKICK icon.

WBAPPICON:

Same story here as with WBDEVICE, it don't seem to be possible for me to obtain specifications on this iconformat, as the defaulticon can't be loaded with the icon.library function "GetDefDiskObject".

Gadgettypes supported:

Complement:

If this option is set, your iconimage will be complemented when picked by the user.

Backfill:

If this option is set, your iconimage will be "backfilled" when picked by the user.

Image:

Set this option if you want your icon to have two different images, one for when the icongadget is up, and one for when it's pressed down. If this option is on upon choosing **Save image**, you'll be prompted to cut out another frame for use as the "selectrender image" (icongadget down). The frame that you've already picked will be used as the icongadget up image.

When you specify the filename of the workbench icon, you don't need to include the .info extension, as this will be done automatically.

If you choose to **Save image** with this option on, the **Save data as** settings will be ignored, as workbench icons are always pure binaries.

1.54 PicCon/Settings/Imageformat/Fontset

This option will let you convert fontsets drawn in ordinary bitmaps to the Amiga's custom font format. This makes it possible to draw fonts in f.ex. DPaint, and then save them as fonts ready to be used in any word processing package, on the Workbench, in your texteditor, in videotitling etc. Currently, PicCon supports only monospaced and monocolored fonts. To learn how this option works, it's best to load one of your existing fonts from disk (set imageformat to Fontset, then choose **Load image**), and then save it back to disk (under another name, in case you should do anything wrong). Just frame the box which contains the data to all your fonts, set the appropriate values in the Options-window and select **Save image**.

The Options-window values should be set as specified here:

XSize:

Set the width in pixels of a single character.

YSize:

Set the height in pixels of a single character.

Baseline:

Set the baseline of the font (must be at least 1 less than YSize). Baseline tells any applications using the font how to handle "low" characters (like g, j, y, etc).

LoChar:

First character in the block you want to save (if this is f.ex. the space, set LoChar to 32 (as the ASCII value for space is 32)).

HiChar:

Last character in the block you want to save (if this is f.ex. the 'Z', set HiChar to 90 (as the ASCII value for 'Z' is 90)).

Note that if your picture consist of several bitplanes, only the bits in bitplane #0 will be saved to the fontsfile, as PiCon does not (yet) support multicolored fonts.

1.55 PicCon/Settings/Imageformat/IFF

Frames saved with this option on will be saved as IFF brushes/pictures (useful for inclusion in drawing packages, DTP programs etc). If you choose to **Save image** with this option on, the **Save data as** settings will be ignored, as IFF frames are always pure binaries.

1.56 PicCon/Settings/Imageformat/SNES modes

If you're not familiar with the **SNES** graphics architecture or if you've just started using PicCon, please check out the explanation of the SNES' different graphicsmodes: **SNES hardware**.

Now for the SNES-mode options:

2-bit characters:

Save data in frame as a 2-bit characterset.

4-bit characters:

Save data in frame as a 4-bit characterset.

8-bit characters:

Save data in frame as an 8-bit characterset.

Mode 7:

Save data in frame as a chunkypixel bitmap characterset.

Swap Hi/Low bytes:

If this option is set and you're saving 2-, 4- or 8-bit characters, the two bytes (representing the bitplanes data) in each character-graphics word will be swapped. F.ex. will a 4-bit character not be saved like this:

1010101010101010

3232323232323232

...but like this:

0101010101010101

2323232323232323

If this option is on and you're not saving an ordinary 2/4/8-bit character screen, but a Mode7 screen, the high and low bytes in the screenposition words will be swapped:

+-----+

|bitmapdata|character| (= 8 + 8 bits)

+-----+

...will become:

+-----+

|character|bitmapdata| (= 8 + 8 bits)

+-----+

Pack equal chars:

Remove duplicated characters from the characterset before saving. You normally wouldn't want to use this option unless you're also specifying to save the screenpositions of the characters (or a MODE7 screen) aswell. This option works with both the ordinary characterbased displays and with the MODE7 characters.

Pack flip chars:

Remove characters that can be duplicated by flipping some of the other characters in the character set. Use this option for maximum optimization (beware that character packing with this option is ~4 times slower than ordinary equal characters packing). You normally wouldn't want to use this option unless you're also specifying to save the screendata as well. The correct bits will be set in each screendata word if this option is switched on (and you're saving the screenposition data along). This option only works with the ordinary characterbased displays (as there's no bits for character flipping in the MODE7 screendata).

Priority:

Use this option to set the priority bit to either 0 or 1 in the screendata words (use this to f.ex. set the priority for SNES-objects to 1, while ordinary screendata have priority 0). This option only works with ordinary non-MODE7 displays, as MODE7 screendata have no priority bit to set.

Save positiondata:

Save out a file containing the screendata positions (characterselection words) along with the character set. The screendata file will be saved to a file with the same name as the filename you specified for the character set data, but with ".screen" appended to the filename. In the ordinary characterbased displays, the screendata in this file will also have the flip- and prioritybits set correctly.

The position data will always be saved in word-format.

With 2-, 4- or 8-bit characters:

+-----+

|y|x|p|pal|characters| (= 16 bits)

+-----+

(y - yflip, x - xflip, p - priority, pal - paletteselection, characters - characterselection)

...and with Mode7:

+-----+

| characters | (= 16 bits)

+-----+

If this option is not set, only the character set which would make up the graphics in your selected frame will be saved, with no information what-so-ever on the individual positions they have on the screen. This will normally only be used with character sets which haven't been packed in any way, or on SNES objects.

(.screen files for character sets that have not been optimized in any way will just be a monotone incrementing row of numbers: \$0000, \$0001, \$0002, \$0003.. while on frames/screens that are packed, the screenposition data will make all duplications and flip-duplications be set correctly.)

Make MODE7 screen:

This option will join, "interleave" and save the characterbitmap data and the screendata positions to the same file, ready to use in a MODE7 display. The bitmapdata will normally be in the highbytes and the characterselection bytes in the lowbytes (opposite if the "Swap" flag is set).

Note that there's no hardcoded limits on any of the saveformats. If you need to save exactly one SNES screen, it's up to you to make sure that the frame you're saving is 256x256 pixels wide. I avoided setting limitations on these saveformats so that PicCon would be as flexible as possible (most often you don't want to save whole screens, as most of the data is just objects or parts of screens, and if PicCon could only save whole screens, lots of storage would be wasted).

There's one exception from this rule, though (regarding flexibility). If you want to make a MODE7 screen and save it, your savefile will always be \$8000 bytes, as characterbitmap data for up to 256 characterbitmaps will always be joined with a \$80x\$80 grid of character selection bytes, no matter how wide and high your selected frame is. The upper left corner of your frame will also be the upper left corner displayed in the MODE7 screen. This is the only way of displaying a MODE7 screen on the SNES. So if you want to make a full MODE7 screen, you have to cut a 1024x1024 pixels wide and high frame, with not more data than can be held in maximum 256 characters. If you cut a frame that is less than 1024x1024 pixels, the rest of the MODE7 screen will be filled up with character number zero in the chunky character set. If this behavior is not wanted, don't set the "Make MODE7 screen" option as the chunky character set and the screenposition data will then be saved to two separate files (as with the ordinary 2/4/8-bit characterbased modes).

If there's anything you didn't quite understand here, it might help to read the [SNES hardware](#) section. If things are still unclear, or if you think there's some stuff I've not implemented that should be included in a SNES converter, please mail (or email) me. Address in the [Disclaimer and Author Info](#) section of this document. (I'm generally not very hard to ask... :)

1.57 PicCon/Settings/Imageformat/SNES hardware

The **SNES** graphicsmodes can be viewed as two separate issues: characterbased and chunkymode. I'll explain the characterbased displaymodes first:

The characterbased displays are made up of screens of 32x32 wordpointers (each word is pointing to a character in a pre-defined character set, and a character is 8x8 pixels wide and high). A SNES screen is therefore $32*8 = 256$ pixels wide and high. Here's a sample screen:

Word pointers

||

TopLeft ||

+-----V-----/-----+

\$0000 |XX XX XX XX XX XX XX XX XX XX XX \ XX XX|

\$0020 |XX XX XX XX XX XX XX XX XX XX XX / XX XX|

\$0040 |XX XX XX XX XX XX XX XX XX XX XX \ XX XX|

\$0060 |XX XX XX XX XX XX XX XX XX XX XX / XX XX|

. \

. |XXXXXXXXXXXXXXXXXXXXX

. \

\$03e0 |XX XX XX XX XX XX XX XX XX XX XX / XX XX|

+-----+

Bottom

Right

The bits in each pointer are used like this:

+-----+

|bit:| F | E | D | C | B | A | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

+---|-----|-----|

| -FLIP- | -P- | -PALETTE- | -CHARACTER- |

| v h | r | | |

| e o | i | (1 out of 8 | (1 out of 1024 available) |

| r r | o | available) | |

| t i | | | |

+-----+

| | |

| | +-

| | These bits are used to

| | select the palettebuffer

| | for this character. (The

| | palettebits will not be

| | used by PicCon in it's

|| current state).

||

| +-

| This bit is used to set priority.

|

+-

These bits choose if the character data should be flipped around the x- or y-axis (or both).

...and the 10 character selection bits are used as pointers into a character graphics table (a character set).

Now...how are the characters stored in this character set? There's 3 different character based display modes, 2- 4- and 8-bit:

A 2-bit character:

+--+--+

lp1lp0 | 2 bytes, 8 + 8 pixels, bitplanes 1 and 0, row 0

+--+--+

lp1lp0 | 2 bytes, 8 + 8 pixels, bitplanes 1 and 0, row 1

+--+--+ .

. . .

. . .

+--+--+ .

lp1lp0 | 2 bytes, 8 + 8 pixels, bitplanes 1 and 0, row 6

+--+--+

lp1lp0 | 2 bytes, 8 + 8 pixels, bitplanes 1 and 0, row 7

+--+--+

p1 - byte in plane 1

p0 - byte in plane 0

A 4-bit character is built with two 2-bit characters; one containing planes 1 and 0, and one containing planes 3 and 2. Like this:

1010101010101010

3232323232323232

(4-bit characters are also used in building up the SNES' equivalent of the hardware sprites of the Amiga (they're called "objects" on the SNES): an object must be 16xYYY characters (YYY is max 512) wide and high (so the SNES-objects have up to 16 different colors, and are $8 \times 16 = 128$ pixels wide). So be aware that to extract SNES "sprites" from PicCon, you have to cut a 128xYYY pixels wide frame in 4-bit character mode).

An 8-bit character is built with four 2-bit characters; one containing planes 1 and 0, one containing planes 3 and 2, one containing planes 5 and 4 and one containing planes 7 and 6. Like this:

1010101010101010

3232323232323232

5454545454545454

76767676767676

(Each number represents the next byte in the respective plane).

And now for the chunkymode (which is also sort of a charactermode (I'll bet you're confused now, huh?)):

In the SNES' version of chunkypixels (called MODE7) the display is made up of characters as well, although the characters now take up $8 * 8 = 64$ bytes, as each character is a small chunkybitmap:

```
+--+--+--+--+--+--+--+
|00|01|02|03|04|05|06|07|
+--+--+--+--+--+--+--+
|08|09|0a|0b|0c|0d|0e|0f|
+--+--+--+--+--+--+--+
|10|11|12|13|14|15|16|17|
+--+--+--+--+--+--+--+
|18|19|1a|1b|1c|1d|1e|1f| (8 * 8 pixels, one byte
+--+--+--+--+--+--+--+ each pixel = 64 bytes
|20|21|22|23|24|25|26|27| each character)
+--+--+--+--+--+--+--+
|28|29|2a|2b|2c|2d|2e|2f|
+--+--+--+--+--+--+--+
|30|31|32|33|34|35|36|37|
+--+--+--+--+--+--+--+
|38|39|3a|3b|3c|3d|3e|3f|
+--+--+--+--+--+--+--+
```

A screen in MODE7 has to be 80×80 characters wide and high ($= 1024 \times 1024$ pixels). Each character on the screen is selected with a byte that indicates the offset into the character set (1 byte = 8 bits = 256 different characters). 256 different characters, each built up with a 8×8 chunkybitmap yields $256 * 8 * 8$ bytes = 4000 bytes. When you set up the MODE7 screen, the 4000 bytes of characterbitmaps are combined with the $80 \times 80 = 4000$ bytes of character selectors to 4000 words ($= 8000$ bytes) of MODE7 screendata. The highbytes of the screenpositions contain the characterbitmapdata and the lowbytes contain the characterselection bytes. So if the first characterbitmap is all ff 's and the first 40 characters of the MODE7 screen are all unique, the first 40 words of the MODE7 screendata would look like this:

```
+-----+-----+-----+-----+-----+-----+
|ff00|ff01|ff02|ff03|ff04|ff05|ff06|ff07|
+-----+-----+-----+-----+-----+-----+
|ff08|ff09|ff0a|ff0b|ff0c|ff0d|ff0e|ff0f|
+-----+-----+-----+-----+-----+-----+
|ff10|ff11|ff12|ff13|ff14|ff15|ff16|ff17|
+-----+-----+-----+-----+-----+-----+
|ff18|ff19|ff1a|ff1b|ff1c|ff1d|ff1e|ff1f|
+-----+-----+-----+-----+-----+-----+
|ff20|ff21|ff22|ff23|ff24|ff25|ff26|ff27|
+-----+-----+-----+-----+-----+-----+
|ff28|ff29|ff2a|ff2b|ff2c|ff2d|ff2e|ff2f|
```



```
+-----+-----+-----+-----+-----+-----+
|ff30|ff31|ff32|ff33|ff34|ff35|ff36|ff37|
```

```
+-----+-----+-----+-----+-----+-----+
|ff38|ff39|ff3a|ff3b|ff3c|ff3d|ff3e|ff3f|
```

```
+-----+-----+-----+-----+-----+-----+
```

...or if the character in position (3,0) on the screen is equal to the one in position (0,0) and the ones in positions (1,0), (2,0), (4,0), (5,0), and so on are unique, we'd get these data for our MODE7 screen (the "Pack equals" option is on):

```
+-----+-----+-----+-----+-----+-----+
|ff00|ff01|ff02|ff00|ff03|ff04|ff05|ff06|
```

```
+-----+-----+-----+-----+-----+-----+
```

```
. . . . .
```

```
. . . . .
```

...and so on (character 0 will be duplicated in position (3,0) and the character in (3,0) will not be saved to our character set).

1.58 PicCon/Settings/Imageformat/Sega

If you're not familiar with the [Sega Megadrive's](#) graphics architecture or if you've just started using PicCon, please check out the explanation of the Megadrive's different graphics modes: [Sega Megadrive hardware](#).

Now for the Megadrive options:

Pack equal chars:

Remove duplicated characters from the character set before saving. You normally wouldn't want to use this option unless you're also specifying to save the screen positions of the characters.

Pack flip chars:

Remove characters that can be duplicated by flipping some of the other characters in the character set. Use this option for maximum optimization (beware that character packing with this option is ~4 times slower than ordinary equal characters packing). You normally wouldn't want to use this option unless you're also specifying to save the screen data as well. The correct bits will be set in each screen data word if this option is switched on (and you're saving the screen position data along).

Save position data:

Save out a file containing the screen data positions (character selection words) along with the character set. The screen data file will be saved to a file with the same name as the filename you specified for the character set data, but with ".screen" appended to the filename. In the ordinary character based displays, the screen data in this file will also have the flip- and depth bits set correctly.

The position data will always be saved in word-format:

```
+-----+
|dly|xl|pal|characters| (= 16 bits)
```

```
+-----+
```

(d - depth, y - yflip, x - xflip, pal - paletteselection, characters - characterselection)

If this option is not set, only the character set which would make up the graphics in your selected frame will be saved, with no information what-so-ever on the individual positions they have on the screen. This will normally only be used with character sets which haven't been packed in any way, or on Megadrive sprites.

(.screen files for character sets that have not been optimized in any way will just be a monotone incrementing row of numbers: \$0000, \$0001, \$0002, \$0003.. while on frames/screens that are packed, the screen position data will make all duplications and flip-duplications be set correctly.)

Depth:

Use this option to set the depth bit to either 0 or 1 in the screendata words.

Note that there's no hardcoded limits on any of the saveformats. If you need to save exactly one Megadrive screen, it's up to you to make sure that the frame you're saving is 320x224 pixels wide. I avoided setting limitations on these saveformats so that PicCon would be as flexible as possible (most often you don't want to save whole screens, as most of the data you want to save is just sprites or parts of screens, and if PicCon could only save whole screens, lots of storage would be wasted).

If there's anything you didn't quite understand here, it might help to read the [Sega Megadrive hardware](#) section. If things are still unclear, or if you think there's some stuff I've not implemented that should be included in a Megadrive converter, please mail (or email) me. Address in the [Disclaimer and Author Info](#) section of this document. (I'm generally not very hard to ask... :)

1.59 PicCon/Settings/Imageformat/Megadrive hardware

The [Sega Megadrive](#) graphicsmode is solely characterbased. This means that the whole display is made up of layers of 8x8 pixels wide characters (that acts as small bitmaps).

The characterbased displays are made up of screens of 40x28 wordpointers (each word is pointing to a character in a pre-defined character set, and a character is 8x8 pixels wide and high). A Megadrive screen is therefore $40 \times 8 \times 28 \times 8 = 320 \times 224$ pixels wide and high. Here's a sample screen:

Word pointers

||

TopLeft ||

+-----V-----/-----+

\$0000 |XX XX XX XX XX XX XX XX XX XX XX \ XX XX|

\$0028 |XX XX XX XX XX XX XX XX XX XX XX / XX XX|

\$0050 |XX XX XX XX XX XX XX XX XX XX XX \ XX XX|

\$0078 |XX XX XX XX XX XX XX XX XX XX XX / XX XX|

. \

. |XXXXXXXXXXXXXXXXXXXX

. \

\$0438 |XX XX XX XX XX XX XX XX XX XX XX / XX XX|

+-----+

Bottom

Right

The bits in each pointer (XX) are used like this:

+-----+

|bit:|F|E|D|C|B|A|9|8|7|6|5|4|3|2|1|0|

+---|---|---|---|---|---|

|d| -PAL- |FLIP- |CHARACTER- |

|e| |v| |h| |

|p|(1 out |e|o|(1 out of 2048 available)|

|t|of 4)|r|r||

|h|l|t|i|l|

+-----+

|||

| +- +-----+

| These bits are used to |

| select the palettebuffer |

| for this character. (The +-

| palettebits will not be These bits choose

| used by PicCon in it's if the characterdata

| current state). should be flipped

| around the x- or y-

+ - axis (or both).

This bit is used to set depth.

...and the 11 characterselection bits are used as pointers into a charactergraphics table (a character set).

Now...how are the characters stored in this character set? This is the format used:

lp - left pixel

rp - right pixel

0,1,2,3 - bit in 0.,1.,2. or 3rd bitplane

.lp..rp. .lp..rp. .lp..rp. .lp..rp. -> 8 pixels in a row

32103210 32103210 32103210 32103210 |

32103210 32103210 32103210 32103210 |

32103210 32103210 32103210 32103210 |

32103210 32103210 32103210 32103210 +- 8 pixelrows

32103210 32103210 32103210 32103210 |

32103210 32103210 32103210 32103210 |

32103210 32103210 32103210 32103210 |

32103210 32103210 32103210 32103210 |

Megadrive hardware sprites are also made up of the same kind of characters. A sprite can be up to 4x4 characters wide and high (sprites can therefore be anything from 8x8 to 32x32 pixels).

1.60 PicCon/Settings/Paletteformat

The settings in this window is for selecting the way the picturepalette will be saved when choosing **Save palette** in the **Project** menu.

Settings in the "Paletteformat"-window:

4 bits

8 bits

32 bits

LoadRGB4

LoadRGB32

IFF

SNES 5 bits

Megadrive 3 bits

Copperlist

1.61 PicCon/Settings/Paletteformat/4bits

Will save out your palette in the standard old chipset and enhanced chipset formats (4-bits-per-gun, 1 word per color).

E.g.:

White is \$0fff, red is \$0f00 and purple is \$0f0f.

^blue component

^green component

^red component

^unused

1.62 PicCon/Settings/Paletteformat/8bits

Will save out your palette in the new AGA chipset format (8-bits-per-gun, 2 words (1 longword) per color).

E.g.:

White is \$00ffffff, red is \$00ff0000 and purple is \$00ff00ff.

^^blue component

^^green component

^^red component

^^unused

1.63 PicCon/Settings/Paletteformat/32bits

Will save out your palette in the internal systemformat (32-bits-per-gun, 3 longwords per color)).

E.g.:

Purple is \$ffffffff00000000ffffffff.

^^^^^^blue component

^^^^^^green component

^^^^^^red component

1.64 PicCon/Settings/Paletteformat/LoadRGB4

Will save the palette as an array ready to be passed to the graphics.library function "LoadRGB4".

Look up the autodocs specification of "LoadRGB4" if you're unfamiliar with the format.

1.65 PicCon/Settings/Paletteformat/LoadRGB32

Will save the palette as an array ready to be passed to the graphics.library function "LoadRGB32".

Look up the autodocs specification of "LoadRGB32" if you're unfamiliar with the format.

1.66 PicCon/Settings/Paletteformat/IFF

Saves out the palette in IFF format (e.g. for reading in DPaint). If this option is set, the **Save data as** setting will be ignored (IFF files are always of binary format).

1.67 PicCon/Settings/Paletteformat/SNES 5 bits

Will save out the palette in the format used by the **SNES**:

E.g.:

Purple is %0111110000011111:

~~~~red component

~~~~green component

~~~~blue component

^unused

Note that the RGB-components have been swapped to BGR (done by Nintendo to get them in the correct alphabetical order...).

## 1.68 PicCon/Settings/Paletteformat/Megadrive 3 bits

Will save out the palette in the format used by the **Sega Megadrive**:

E.g.:

Purple is %0000111000001110

^^^^^unused

^^red component

^^green component

^^blue component

Note that the RGB-components have been swapped to BGR.

## 1.69 PicCon/Settings/Paletteformat/Copperlist

Will save out your palette as a list of commands that the Amiga graphicscoprocessor (a.k.a. COPPER) understands. Insert the palette coppercommands into your own custom copperlist to make use of the paletteentries.

PicCon knows the format of both OCS/ECS and AGA chipsets, and will save in the appropriate format according to the number of bits-per-gun specified:

4 bits-per-gun: OCS/ECS copperlist.

8 bits-per-gun: AGA copperlist.

The other options (32bits, LoadRGB4, LoadRGB32 and IFF) will ignore the Copperlist setting.

## 1.70 PicCon/Settings/Grid settings

Imagine this: you are coding a game where you need loads of background graphicstiles (say, 16x16 pixels wide, which are then combined to make bigger objects in the background). To get maximum speed (minimum CPU/blitter load), you need to have each one of them in interleaved RAW format. What do you do? You'd probably save out the whole bitmap with an IFF-converter and then code your own custom formatconverter to get the tiles the way you wanted them. This is not necessary any more, as PicCon is able to save a grid of frames in any of the usual formats, except as IFF ILBMs or as Workbench icons.

The options in the grid settings window are explained below:

Grid orientation (horizontal/vertical):

Choose to save the tiles in either left-to-right (horizontal) or top-to-bottom (vertical) fashion.

(As an example, let's say we've got a 3x3 grid of frames:

```
+---+---+---+
|   |   |   |
| 1 | 2 | 3 |
|   |   |   |
+---+---+---+
|   |   |   |
| 4 | 5 | 6 |
|   |   |   |
+---+---+---+
|   |   |   |
| 7 | 8 | 9 |
|   |   |   |
+---+---+---+
```

With horizontal orientation, these frames will be saved in this order 1,2,3,4,5,6,7,8,9, while if you set vertical orientation, the internal positions in the saved file will be like this: 1,4,7,2,5,8,3,6,9.)

Blockwidth:

Set the width in number of pixels in the individual tiles.

Blockheight:

Set the height in number of pixels in the individual tiles.

Blocks horiz.:

Number of blocks horizontally. (3 in the above example)

Blocks vert.:

Number of blocks vertically. (3 in the above example)

Total #:

The total number of blocks you want to save (if no blocks are "missing" in your chunk of frames, this number will usually be (blocks horiz.)\*(blocks vert.)). (9 in the above example)

Note that your chunk of frames to be saved must have the upper left pixel in the absolute upper left corner of the screen (at (0,0)).

The gridsave option will not be influenced by any box you've cutted in the picture. The individual tiles will all be saved in the format you select in the **Imageformat** window. All tiles will be appended together in a single file.

See also **Save grid**.

## 1.71 PicCon/Settings/Grabpen

Choose a new boxpen to search for in the **Grab frame** function.

## 1.72 PicCon/Misc

Menuchoices on the "Misc"-menu:

Pen usage in frame

Remap pens in frame

Trace unused pens

Convert EHB->normal

Reselect systempens

Fake SNES screen

Remake Amiga screen

## 1.73 PicCon/Misc/Fake SNES screen

Due to a different videoarchitecture, pictures ported directly from the Amiga to the **SNES** (or the other way around) will look a bit strange. This is caused by the fact that while the Amiga (in PAL mode) uses 320x256 pixels to build one ordinary low resolution screen, the SNES uses 256x256 pixels to make the exact same display. So while the SNES has a 'pixelratio' of 1:1, the Amiga has a ratio of 5:4. The "Fake SNES screen" function is supposed to simulate the way your graphics will look on a SNES display. This is achieved by first duplicating all pixels 5 times horizontally and then 'divide' the pixelwidth of all pixels horizontally by 4 by setting the SuperHighRes mode. By doing this, we'll get the same pixel aspectratio as the SNES does. Clever, huh? Thanks to Daniel Hansson for coming up with this excellent idea!

To use this function, your picture has to be displayed in lowres, non-interlaced, non-HAM on a PAL screen (otherwise, the aspectratio would still come up all wrong...). If your picture is not in this format, and you still want to use this function, you may change the displaymode with the **Change screenmode** option in the Project-menu.

Don't save any graphics from this display for use on the SNES, as this display will be treated the same way as any other PicCon display (that is; all pixels would be saved 5 times horizontally). You may wish to save out the picture as IFF ILBM in this new format for further editing in DPaint, though, as you can infact draw in this 'faked' SNES displaymode by using a 5x1 pixel brush (and only insert the 'SNES-pixels' on a 5x1 grid). Later you can then use the **Remake Amiga screen** function to obtain an ordinary Amiga display (and then save out the graphics in the SNES RAW modes).

## 1.74 PicCon/Misc/Remake Amiga screen

Before you read this, you might want to read about the **Fake SNES screen** function.

Use this function to convert a faked SNES display back into the ordinary Amiga display. This is done by discarding 4 out of the 5 even pixels and resetting to the low resolution display. Note that this function will also work with ordinary PAL SuperHires Non-Interlaced screens, but will in that case just partly destroy your picture.

## 1.75 PicCon/Misc/Pen usage in frame

This function makes a histogram of the relative usage of the pens in the selected frame.

---

## 1.76 PicCon/Misc/Remap pens in frame

I guess this function was pretty high on the "most wanted feature" list by the coders? (It's not included in DPaint, and the version of this function implemented in Brilliance can't work on selected areas of the screen.)

Anyway, it gives you the possibility of either moving or interchanging all pixels of a pen to another pen. This function works inside your selected frame.

Do not confuse the "Move" and "Swap" functions in this feature with the corresponding calls in the "Modify palette" option. Nothing is actually done with the values of the paletteentries, but with the bitrepresentations of the individual pixels in the bitmap.

See also the quick tutorial: [Remap pens in frame](#).

## 1.77 PicCon/Misc/Trace unused pens

This function will find all the unused and duplicated pens in the palette and "push" them at the end (by moving all pixels set with other pens into the front of the palette). If the number of unused and duplicated palette-entries are so high that 1 or more bitplanes are not needed (say, if you only use 13 pens in a 5 bitplanes picture (which has 32 pens available), it would be possible to render the exact same picture in 4 bitplanes), you'll also be asked if you want to remove the superfluous bitplane(s).

See also the quick tutorial: [Trace unused pens](#).

## 1.78 PicCon/Misc/Convert EHB->normal

This function converts a 6 bitplanes picture in extra halfbrite modus (with 32 editable paletteentries) to an ordinary 6 bitplanes picture (with 64 editable paletteentries).

This also makes it possible to use the other simple "imageprocessing" tools in PicCon to work on these pictures.

## 1.79 PicCon/Misc/Reselect systempens

If for any reason the selected systempens (the pens which are used to render the menubar, requestergadgets etc) should make something hard to read, use this function to reselect them.

(The systempens might get changed if you use the paletterequester, and you should then use this function if they get hard to read (or just ugly to look at). This also applies if you use the [Load palette](#) option).

Be aware, though, that on Amiga's with OS2.0 it's impossible to control the menubar pens.

## 1.80 PicCon/History

History:

.  
.  
.  
|  
|

v1.07:

- fixed many small bugs and quirks (as usual)

---



- now perfect handling of the smallest bitmaps (e.g. DPaint-Brushes)

v1.08:

- included option to save data as linkobject
- included loading of raw- and IFF-palettes
- option to save palette as IFF was added
- new imagesave format; IFF
- raw savefiles now have frameinformation in the commentfield
- loading of RAW images and WorkBench icons implemented
- now possible to save multiple sprites in one go

v2.00:

- removed bug that sometimes caused crash upon exit
  - paletterequester
  - option to iconify PicCon added
  - fixed a bug that made it difficult to specify memorytype and external definition name on palette linkobject files
  - no longer needs datatypes.library to be present
  - IFF ILBM loading without datatypes.library
  - IDCMP\_VANILLAKEY - no need to hold right Amigakey when using hotkeys any longer (more like DPaint ;-)
  - now saves WBicons at any width (don't have to be multiples of 16 pixels)
  - fixed icontype bug on icons that use drawerstructures
  - no longer needs iffparse.library to be present
  - added option to save frames with extra blitterwords attached either left or right (or both) for easy blittershift operations
  - added option to save chunky images in packed format
  - added option to load raw chunky images in packed format
  - quick automatic framegrabs with less userinteraction is now possible ("Grab frame")
  - image and palette saveformats displayed in titlebar at all times, along with the coordinates of the framebox
  - new saveoption: Grids (nice for saving e.g. 100s of blocks of graphical data for maptiles/spriteframes/...)
  - functions to compress pens and remove superfluous bitplanes added
  - conversion: EHB --> 64 colors normal
  - added control of systempens (all menus/requesters/gadgets should be fully visible on all but the most extreme palettes (like all paletteentries having e.g. the same value))
-

- new autoscan feature
- added option to save data as Pascal sourcecode
- made more functional demoversion

v2.01:

- fixed a bug in the demoversion

v2.10:

- better (more flexible) imagesave options input
- AutoScan now also works with other saveformats than pure binary
- added new palettesave mode: SNES 5-bits
- added new RAW save formats: SNES characters and SNES Mode 7
- added "Fake SNES" and "Remake Amiga" functions to help SNES developers
- made better algorithm for system penselection
- fixed a bug in the linkobject format
- made AmigaGuide docsfile more fontinsensitive (reformatted text and turned on autowrap)
- added a new Settings-window for editing of general settings in PicCon (for removal of annoying requesters, etc)
- removed all OS3.0 dependencies. PicCon no longer requires OS3.0+, but OS2.04+ (and still takes advantage of 3.0 features)

v2.20:

- removed a bug that caused PicCon to still be incompatible with 2.0
  - automatic processing of pictures from command line (makes it possible to have scripts that does the whole job for you, without even entering the program!)
  - made "Settings"-menu available on startup
  - made "Save prefs" and "Iconify" options available on startup
  - removed separate "Load palette"-settings window (now possible to load all kinds of palettes that are saveable from PicCon)
  - removed separate "Load image"-settings
  - added pattern gadget to filerequester
  - added the option to load and save Amiga monospaced, monocolored bitmap fonts
  - fixed a bug with HAM8 bitplanes/interleaved format
  - included support for saving as E source
  - fixed a bug in the Pascal source saveformat
  - Sega Megadrive character set saving
  - Sega Megadrive palette saving
  - fixed a small bug in "Trace unused pens"
  - fixed a bug in SNES character flippacking
  - made it possible to move boxwalls with cursorkeys
-

## 1.81 PicCon/Thanks to...

Thanks to...

- \* my girlfriend Guri
- \* my brother Mads (for doing excellent graphics)
- \* Stian (for nice coding)
- \* Mika Saastamoinen for being my first registered user :)
- \* my betatesters (Vidar Petursson, Børge Nøst and Daniel Hansson (special thanks to Daniel Hansson for lots of help with the **SNES** implementation))

... and to everybody else who has helped me with bugreports and clever suggestions for enhancements, and last (but certainly not least): those who have registered.

---