Multiformatinputfilesorrydoesn’tknowthisargumentisnotusers

COLLABORATORS			
	TITLE : Multiformatinputfilesorrydoesn’tknowthisargumentisnotusersfault.		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		July 20, 2024	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Multiformatinputfilesorrydoesn’tknowthisargument

1.1 Madhouse 2.01 Contents

```
Have you got a screen blanker?

Yes? Here is another one. The main-program, Madhouse, is *VERY* confi-
gureable and has a lot of options. But look at them yourself. To see all
the effects and to start working with Madhouse, read the little Workshop.
Madhouse has to offer some good blankers, but lost some during troubles
with AMOS. In the near future, we will have a lot of more blankers.

Now enjoy!

              What you need: The necessary Amiga.
               The Workshop: see the blankers!
          MadhouseConfigEd: all pages.
                  Madhouse: Tooltypes.
                     Error: Madhouse Error-Messages
   Madhouse for Programmers: own Modules
                The Blanker: and all Options
                 The Autors: come on, read this stuff
```

1.2 What you need for Madhouse

```
Nowadays, even screen blankers have special needs. Here is what you need
for Madhouse:
    o An Amiga with OS 2.0, localized (but still english in your case :-)
      with OS 2.1.
    o A hard disk, where you should install Madhouse. The newer versions
      of Madhouse need, unlike Madhouse 1.1, a hard disk.
    o MUI (© Stefan Stuntz). You won't need the brand-new version 3.1,
      older ones do well, too. See MUI-Info.
    o Madhouse was programmed to use as less memory as possible. That's
      the reason why 'Madhouse' has no graphical user interface, but a
      seperate program, 'MadhouseConfigEd' does all the configuration.
      'Madhouse' has only a size of 30 KByte.
```

1.3 The Workshop

All right, you have pressed the right button to have fun with Madhouse.
Now, we explore the functions of Madhouse together.

Installation

The installation can be done completely by the Installerscript, which is part of this distribution. All you have to do is double-click on the icon named "Install". But this will only work if the program "Installer", which executes the script, is in your C: directory. If this is not the case, you can search it on your "Install"-Disk (if you have one) or on the disks of other programs which submit an Installerscript, and copy it to C:.
If you have no possibility to get the program (I don’t expect so), then you have to install Madhouse "by hand".

- For a quick test of Madhouse you can start Madhouse out of its original directory, if you don’t remove the floppy disk / CD Madhouse is on (that’s why you should install Madhouse on HD, it always needs it’s blankers...)
- For permanent usage: as the word "permanent" sais, the WBStartup-Drawer will be used. So copy the program "Madhouse" to SYS:WB-Startup. All programs in WBStartup are started everytime you boot your computer. Make a drawer on your hard disk, and copy all Madhouse files into it. The background-process "Madhouse" will start the MadhouseConfigEd itself, if this is necessary or if you want so. Don’t start it yourself, but make a start by Madhouse possible. So select the icon of the Madhouse in your WBStartup-drawer, and call the menuitem "Icon/Information". Change the Tooltype "CONFIGED=": insert the complete path and the name of MadhouseConfigEd after the "=". Do not use spaces. If your ConfigEd is in Work:Madhouse, the line has to look like this: CONFIGED=-Work:Madhouse/MadhouseConfigEd You don’t have to copy any catalogs, because "English" is the built-in language of Madhouse.

Start the program.

This should be easy. Double-click on the icon.
If you don’t have OS 2.0, Madhouse is not the only reason to upgrade. In other words: OS 2.0 is needed. And a BIT of memory.

A window will be opened.

If not, everything is correct. If you want to open the window (which means that you want to start MadhouseConfigEd), just select ‘Madhouse’ in the Tools-Menu.
But at the moment, Madhouse cannot find it’s ENV:Madhouse.prefs, and calls the ConfigEd to do the settings.
If you get a requester which tells you that Madhouse couldn’t find the MadhouseConfigEd, click on the gadget. Because it has no preferences, Madhouse won’t be able to run, and quits. Now you can correct the path to MadhouseConfigEd in the Tooltype of WBStartup/Madhouse, and run Madhouse again. Now anything should be fine, and Madhouse should start the Con-figEd.

Now enter the path of the blanker-modules!

Somewhere on your hard disk, you must have installed a directory named ‘blankers’. Next to the string gadget ‘Path’, there is a little Popup-Button which opens a file-requester. Click on it, and select the path which includes this blankers-directory. After that, you should see names like "FlyingToasters" and so on. Be sure to select a path which begins with the name of the disk - "Blankers/" is wrong, but "Work:Madhou-se/Blankers" will do fine. Click on ‘Okay’.
If your directory was the right directory, the gadgets "Use", "Cancel" and some others are enabled.

Now we want to see the blank-Modules

Just click twice on a name in the list. This will open the BlankerPrefs-window. Please click on "Test". Great, isn’t it? To abort, press a key or a mousebutton. Now change the parameters with the gadgets above. Click on "Cancel" to exit and on "Okay" to exit and save. Choose other blankers in the list and have fun.... (some minutes later) ...finished? If you like Madhouse take a closer look at the other gadgets in the mainwindow, otherwise click on remove. If you are missing blankers you’ve seen in other

More information about the list.

The list on the left side of the window has three functions. Blankers which have their checkmark enabled will be used for blanking. Go on the Blankers-page, and you will see that one click on a name of a blanker will display information about him. And you know already what does happen when you double click on a name, don’t you?

Select the time after the blanker should start to blank!

With the "Time"-gadget.

Save your configuration.

Nothing easier than that: Click on "Save". The configuration will be saved as "ENV:"- and "ENVARC:Madhouse.prefs".

1.4 MadhouseConfigEd

The three gadgets ‘down under’
The List
System - Page
Options I - Page (Other Options)
Options I - Page (Dimmer)
Options II - Page
Blanker - Page
Info - Page

1.5 Import Function

The Import function of Madhouse mades it finally possible to use Garshne- Swaz- and Madhouseblankers in ONE program as they were of the same protocoll and format. Perhaps will other programmers of modulary Screen-savers stop converting every source code they can get to their own system now.

To Import other blankers, you'll have to get them first. We respect the copyright of other authors and don't submit their blankers directly. So get an original Swaz- or Garshneblanker archive (or an extension package) and extract the files. Here are some Aminet filenames, which are

- interesting:util/blank/AgBlankers1_1.lha
- util/blank/ARoseGarshn1_4.lha
- util/blank/GBlnk36.lha
- util/blank/GBlankM1.lha
- util/blank/GBlankM2.lha
- util/blank/NoGBlank_1_1.lha - (not that good)
- util/blank/SBTimeModule.lha
- util/blank/ShowPics.lha - (to show pictures with the Garshneblanker 'Executor')
- util/blank/SwazBlanker27.lha

Perhaps I should mention that I have no Aminet access at all, I buy these Aminet CDs. That's why it could be that some version numbers are already old. I want to thank Martin Schulze, an Aminet administrator, who copies our Madhouse archive into the Net.

Please notice that Garshne- and Swazblankers need their own library, which is NOT submitted with extension packages. So begin with importing the contents of SwazBlanker27.lha or GBlnk36.lha.

Ok, is your archive decrunched? Now click on one of the Import buttons, depending on which type of blankers you want to import. A window opens.

Hints

You should read them, too.

"ImportHelp"-Directory

I'm sure you have noticed a directory with that name in the Madhouse-disribution. Just select it's path in this string gadget. The directory contains files like 'Garshne.Clock', which means that this file contains further information about the Garshneblanker 'Clock'.
These additional information is needed, because other blanker's don't have functions which are in Madhouse. An example is 'CPU active', Garshneblanker is CPU sensitive, but it starts no blanker at all when a program is calculating heavily. Madhouse would try to start a simple blanker, but how should it know which Garshneblankers are simple? (In Madhouse, this information is noted in the 'gadget'-file of every blanker.) This and more information is inside this ImportHelp-directory.
Another example: if you do an Import without selecting this directory, you will get 'Unknown' in the cycle gadget 'Shows WB' on the Blanker -page for every new blanker, instead of 'Yes' or 'No'.

Blanker Source

is the most import gadget. Select the directory where the blanker modules are. Perhaps you need to rename some blankers, if they have a cpu-specific extension.

blanker dest.

I think you agree that the new blankers belong in Madhouse’s blankers-directory...

Library-File source

If you are not installing an extension package, you should select the path + filename of the blanker’s own library, like swazblanker.library. Madhouse will just copy it to Libs:. Be sure to install submitted libraries, I have no idea what the new blankers will do without their library...As explained in the Hints for Swaz, Swazblanker has two libraries. You must install matrix.library, too. You can do this with a directory tool, or you can Import a second time and select nothing but the other library.

Documentation source - Swaz only!

Swazblankers do their configuration alone. The programs have a menu item called ‘Help’, which shows the AmigaGuide-Documention for the blanker. If you select the drawer ‘docs/blankers’ from the original Swazblanker archive, Madhouse will scan it and will place the doc files in the new blanker subdirectories, which are created for the new blankers as well. Now, this Help function should work.

Documentation dest.

See above...

Progress

Here you can watch Madhouse processing the new blankers.

Okay and Cancel

Okay starts the operation, Cancel closes the window.

How to update imported blankers

There is of course the possibility that blankers which you imported once, get updated. In this case, just import the updates like you do it normally. Madhouse will notice that there are blankers which have the same name and are of the same ‘type’ (are reached with the same protocol). You will be asked whether you want to replace the old blanker, or if you want to skip this blanker. For updating purposes, you should select ‘Replace’.

Please notice,

Madhouse is Shareware. You may configure and start the imported in MadhouseConfigEd, but Madhouse will refuse to start them. You need a

keyfile, which you can get from the authors for DM 20 or US$ 15. We think that price is okay.

1.6 The List

With Madhouse 2.01, the two lists were replaced by one. But the function hasn't changed. These are the things you can do with the list:

- The little checkmark gadgets next to the names of the blankers are used to define which blankers will be started by Madhouse.
- With a single click on a blanker name, information about this blanker is displayed on the Blankers-Page. Some internal settings of this blanker are enabled on this page, too.
- Try a double click on a blanker name! This will open the Blanker-Prefs-window of the blanker.

I hope you like the new design of the listview. With more than sixty blankers in the list, it's now easier to find a blanker...

1.7 The three gadgets `down under'

Save

Closes the window and saves all settings, but not the ones in the Blanker-windows (there, you have an extra Okay-button) to ENVARC: and ENV:-Madhouse.prefs.

Use

Closes the window and gives the new settings to Madhouse, but doesn't save them. After a reset, the settings that were saved using 'Save' will be present again.

Remove

Quits Madhouse and MadhouseConfigEd.

And how can I 'Abort'?

Madhouse will keep running, but will not using the new preferences you set in MadhouseConfigEd, if you just close the window of MadhouseConfigEd.

1.8 System - Page

If everything is 'disabled' on this page, read the Workshop.

Import-Buttons

The buttons and the windows which belong to them are explained in

Import-Function

Path

If the Use-gadget and some others are disabled, use this gadget to enter a correct path for the blankers.
Be sure to insert a complete path, with the name of the volume in it. The path is correct if it ends with "blankers" (if you haven’t changed the directoryname). Madhouse recognizes the path if it has the file "the_-right_drawer" in it, so don’t delete this file.

Blank Time

If you leave your Amiga, Madhouse waits a special time before its starts the first blanker. Select this time with the ‘Blank Time’-gadget (in seconds).

Dimm Time

Here, you get set whether you want to use the dimmer or not (Checkmark-gadget), and after which time it should start (Slider). The dimmer has to work before the blanker, not the other way round. So you should set a shorter Dimm Time than Blank Time, with some seconds (10 to 50) difference.

Text viewer

In some cases, Madhouse will need your favorite text viewer to display the error messages of the blankers. They rarely give you errors, but if they have one, Madhouse uses the text viewer to display them. This function was designed to save code (and memory!) in the ‘Madhouse’-program, instead of building an own user-interface.

Change Blankers

With this gadget you can specify if Madhouse changes the blankers after a while. If you turn it on, the "Duration"-Slider on the Blankers-Page will be enabled. With the "Duration"-Slider of every Blanker-module can be selected how long it should work.
Of course it is not very intelligent to change the blanker if only one is activated, in this case the "Exchange Blankers"-gadget will be disabled.

1.9 Options I - Page (Other Options)

CPU active

The first "other option" is the "CPU active"-setting. With this cycle-gadget you can select what Madhouse should do if your computer has something to work on. To start a complex screen-blanker would slow down the calculations. So Madhouse checks the CPU-activity and - if your processor is working - uses this setting to decide what to do. The cycle-gadget has three positions:

- use all Blankers means that Madhouse doesn’t take care of your CPU at all.
- only simple ones influences the random-function of Madhouse. If

this option is active, Madhouse will only start Blankers which hardly need the CPU. If you selected only Blankers which use your CPU a lot, you will see a blank screen and after that a message which tells you about this situation.
- show nothing tells Madhouse that it has to open a black screen if your computer is working.

If you want to know which blankers are the big CPU-users, take a look at the Blankers-Page.

Sound

This gadget switches between the two ways of playing mods while blanking:
- with the medplayer.library by Teijo Kinnunen, which just plays MED or
- via Delitracker. Madhouse controls DeliTracker (© Delirium Softdesign, Peter Kunath and Frank Riffel) with its ARexx-Port. You don’t have to start RexxMast for this, RexxMast is needed if you want to run ARexxMacros, but not for the communication of two "real" programs.

1.10 Options I - Page (Dimmer)

Okay, here comes the dimmer. It only works with OS 3.0 or higher, because only 4096 colors would look terrible for a dimmer.
A dimmer dimms (fades) the screen, together with the ‘Dimm Time’ Slider on the System-Page, you can be ‘warned’, before the blanker starts. Anyway, the dimmer looks nice.

Delay

With Delay, and with

Step

you can control how fast the dimmer will work. I advice you to let one of the sliders be 1. Step gives Madhouse the amount of color-steps it goes in one step; and delay is the time in 1/50 seconds which is between two steps. Try it out...

Kill WB-Copperlists

Programs like WBVerlauf (© Christian A. Weber) give the WorkBench a nice copper list. But if you dimm the colors, the copper list will stay as it was. With ‘Kill WB-Copperlists’ enabled, Madhouse will remove the copper list and install it again, after dimming.

Depth red, green, blue

Normal dimmers have a special depth, ‘how far’ they dimm. In Madhouse, you can control this depth seperate for red, green and blue. The higher you set a slider, the less color will remain. An Example: If you set Red=20, Green=240 and Blue=250, you will get a red horror-Workbench while dimming. Experiement with these sliders, you will find alot of nice and ugly compositions!

```
Undimm speed

Normal dimmers undimm at once, after you move the mouse. You can reach
this behaviour in Madhouse by setting ‘Undimm speed’ to the maximum. But
if you set smaller values, the dimmer will undimm slower.
```

1.11 Options II - Page

```
Immediate Start

With this options, you can select Madhouse’s reaction on you moving the
mouse cursor in an edge of the screen. This reaction can be:
    o No blanking. Is the mouse cursor in the selected edge, Madhouse
      won’t blank, no matter how long you do nothing. The dimmer won’t be
      started, too.
    o Dimmer. This will start the dimmer immediately, but won’t blank
      after the selected time.
    o Blanker. This starts the blanker immediately.
With the three cycle-gadgets, you can give every action an edge, or you
don’t use the action (‘Off’).

Password

If you want to use a password, check the ‘Use’-Button. Enter your pass-
word in the string-gadget. BE CAREFUL: Write it under your mouse or on a
disk or behind your monitor, but without knowing your password, there
will be no chance to go back to your application. (I hope so...)
After setting the password, Madhouse lets you enter it again to see if
you are able to remember your password, and if all used chars are valid
for a password (you cannot use <Alt>, <Ctrl> and the Amiga-keys).

After you stopped a blanker, the password-screen will appear. You see
only a string-gadget (which is, in fact, not a real one). You don’t see a
curser, and all chars you enter will be only displayed as a "*". You have
three chances to type the correct password. You can abort the process by
pressing Escape or confirm the password with Return. Backspace deletes
the right-most char. Other keys are not supported. Case is sensitive.

Please check carefully if the Password-Screen is _really_ safe on your
computer-configuration. After a wrong password, it can take Madhouse a
short time to start the next blanker.
```

1.12 Tooltypes of `Madhouse’

```
Madhouse itself supports two tooltypes:

CONFIGED

Type the complete path of MadhouseConfigEd after the "=", example:
"CONFIGED=Work:Madhouse/MadhouseConfigEd". Do not use spaces next to the
"=".
```

QUIETQUIT

If this tooltype is set, Madhouse does not bother you with a requester, if you want to quit it by starting it again. Remove it or set it in brackets to enable the requester.

1.13 Blanker - Page

Please notice

that all settings on this page (excluding the BlankerPrefs window) will be lost if you don’t click on ‘Save’ or ‘Use’. If you read the blankers directory again, using the ‘Path’ gadget, they will be lost, too.
The active or highlighted blanker means the blanker which has its name written on a colorful background, after you have clicked on the name once or twice.

Duration

With this slider, you can select how long the highlighted blanker will work, after this time (in minutes) Madhouse will start another blanker. Note that this will only happen if ‘Exchange Blanker’ is selected. Otherwise, you cannot use this gadget.

Author

Here you can see the name of the author of this blanker.

CPU load

Here you can see if the blanker uses the CPU a lot ("medium to high") or if he hardly uses it ("low"). This is necessary for the "CPU active"-setting, which will not take a "medium to high"-blanker if "use simple ones" is selected there.
See CPU active for further explanation.

Version

Here you can see the version-number of the selected blanker.

Task Priority

sets the AmigaDOS exec priority of the blanker task. You should use this function with care, and select ‘0’ for most of the blankers.
But I give one example where ‘1’ is useful: for BouncingPixel. If the CPU is active (a program is calculating), and you have not selected ‘show nothing’ in that cycle gadget, Madhouse will run a simple blanker (if you have selected ‘only simple ones’; you should not select ‘use all blankers’). This simple blanker may be BouncingPixel, because it shows ‘low’ in the ‘CPU load’ field. Together with the calculating program, the pixel won’t run that smooth. But with a Task Priority of 1, it will run smoothly, and the calculating program won’t be slowed down.
Because BouncingPixel is a very simple blanker. Here you can set higher priorities than ‘0’.

Protocol

The type of the blanker you have selected.

Shows WB

Some blankers copy the frontmost screen and do some things on it (Shuffle, Worms and others). This is meant with ‘Shows WB’.
With ‘Exchange Blankers’ on, Madhouse will stop a blanker and start a new one after ‘Duration’ is reached. For some secounds, you would see the Workbench, but Madhouse opens a little black screen, so that blankers will change more ‘quietly’. Of course, copying the frontmost screen will end up in Shuffle shuffling this little black screen...
For this reason, you can select ‘Yes’ in the ‘Shows WB’ cycle. Madhouse will simply close it’s little black screen before starting a blanker which has the attribute ‘Shows WB’ = ‘Yes’.
‘Unknown’ does only appear if you have imported without using the ImportHelp-directory. This is treated as ‘No’.

Delete Blanker

If you imported a lot, you’ll notice how often authors of modulary Screen Savers copied their blankers. Some look even more equal than others ;-).
With this button, you can delete the active blanker, not only in the list, but also on your hard disk!

Take Duration for all Blankers

If you don’t like the pre-defined 5 mins., you can select the time you want in the Durating setting of the active blanker, and than click on this gadget. Now every blanker in the list has this setting.

Sound

With this sound-menu, it is possible to assign a music-module to a blanker and to play it.

For this purpose, the medplayer.library or the DeliTracker is used, depending on what you set on the Options I - page. If you want to use the DeliTracker, it has to be running; if you want to use the medplayer.library, you need it in LIBS: and you can only use MED-mods.

First of all, you’ll have to select a blanker (click one time). If you already selected a mod for it, you will see it in the string-gadget. Of course you have not, otherwise you won’t read this chapter... So click on the popup-button next to the string-gadget.
The Sound-Popup appears. It has the following gadgets:

Add...

Use this button (and select a filename) to add a file to the soundlist.

Del

Deletes a file from the soundlist. Every blanker which had this mod in his sound-setting has now no module any more.

Tapedeck-play-arrow

Do you really need an explanation for this gadget? Arrrgh, here it is:
this gadget plays the selected module.

Tapedeck-stop-button

Stops the music.

The soundlist

The task of the soundlist is to handle the modules you want to use with Madhouse. You can double-click on the list to use the selected mod with the selected blanker and to close the popup. But the real intention for me to program this list was the fact that every user expects a list when he or she clicks on an popup-button...

By the way, you can enter the pathname directly into the string-gadget, but that’s boring, isn’t it?

If you think that Madhouse saves the list itself (your hard work!), you’re wrong. Madhouse only saves the blanker-sound-settings and will rebuild the list on the next start of the program. So if you add a module to the list and if you don’t use it, it will be erased after the MadhouseConfigEd quits.

Now here is a nice hint from Aicke: if you want the Delitracker to be quiet after Madhouse-activities, disable ‘Play at Start’ in the Delitracker-Config. And it’s a good idea to turn off ‘Songend’, too. Then, Delitracker won’t load a new Soundmod when the current blanker blanks longer than the selected mod for this blanker can play.

1.14 Info - Page

Here, you can read a lot.

1.15 Madhouse - Error Messages

Madhouse knows a lot of errors... The errors are devided into two groups.

One the one hand, the main-programs can display errors. You see them in a window on the Workbench, with a "Continue"-gadget below.

On the other hand, the blankers check their environment and the selected options. If something goes wrong, they don’t display the error themselves, but they give the text to Madhouse. If you started the blanker yourself in the BlankerPrefs-window, the window will get a bit larger, and you can read the error in the bottom of the window.
If the blanker was started by Madhouse, Madhouse will first try to start other blankers you selected. If no blanker is left, you will see a black screen. You can close the screen like you quit the blankers: by pressing

a key or a mousebutton. After that you see your text viewer, displaying a text about the errors which were reported by the blankers.

If you want more information about a blanker-error you can look at it’s description.
The errors from the main-programs are explained below (excluding these ones which have to do with "not enough memory"-situations).

File ENV:Madhouse.prefs does not exist.

This is not a real problem, it simply shows you that you have to configurate Madhouse again, because it’s config-file "ENV:Madhouse.prefs" is not available.

Please select a COMPLETE path

Here you have to input a path with the name of the volume in it, like "Work:Tools/Madhouse/blankers".

BlankerInfo-Chunk does not exist!

In the "gadget"-file of this blanker is no BlankerInfo-Chunk. So Madhouse cannot read whether the blanker uses the CPU a lot or not.

Unknown macro: xx

In the gadget-file of the called blanker is a macro xx which doesn’t exists.

Couldn’t create the subdirectory "Ram:..."

Perhaps your Ram:-Disk isn’t mounted.

You started Madhouse the second time. Do you want to remove it?

To start Madhouse twice is another possibility to quit it. This requester asks you if it is what you wanted.

Couldn’t load the "gadget"-file!

The gadget-definition is needed to open the window.

The "prefs"-file is not in the subdir!

Not a real problem. You will have to configurate this blanker again. Move every slider, on the startup without a prefs-file in the blanker’s subdir every slider has the value 0.

Couldn’t access "RAM:Madhouse_Storage/prefs"!

There is no prefs-setting-file in the blanker’s subdirectory.

Couldn’t start this blanker:

The "blanker"-file in the blanker’s subdir is missing.

Problems with your password

You cannot use <Alt>, <Ctrl>, <Caps Lock> and other keys while entering the password.

Need a stack minimum of 4.096!

The stack is just a block of memory which is allocated by DOS (not by the program) for every started program. So the program itself has no ability (on my knowledge) to size this stack. The user who starts the program has to size this stack. But don’t worry: normally it’s easy and you don’t have to do it often. Where you input the stack size (which should be 4096 Bytes) depends on where you start Madhouse. I explain three ways here:

- From the Workbench or by moving Madhouse’s icon into the WBStartupdrawer: Click once on Madhouse’s icon. Select "Information" in the "Icon"-Menu. A requester appears. Click in the box near "Stack:" and correct the value.
- From the Shell or the S:User-Startup / S:Startup-Sequence: Every Shell-window has it’s own stack-value. The programs you run from this Shell get this value. You set this value by typing stack 4096 <Return> in this window before starting Madhouse. In the Startup-Sequences you insert this line before the Madhouse-call.
- From the Toolmanager: Enter the value 4096 in the program-requester for Madhouse.

4096 bytes is the normal value for Amiga-programs - so there shouldn’t be any problems, but who knows?

Bug #1 in program!

It’s hardly possible to get this error. The Bug #1-error is only a help for me to write a program with less/no bugs.

No chance to run Madhouse on your 1.3-machine!

Arrrrgh! OS 2.0 is needed by so many programs, why do you use this terrible 1.x-version? You can get the new ROM’s and the software in a lot of Amiga-shops, and if your old software needs 1.3 you can buy a switch for two ROMs. It’s very easy to insert the new ROM’s into EVERY Amiga, and it doesn’t cost much!
If you have OS 2.0 and you get this message, you have a real problem...

Couldn’t open asl.library V37!

The asl.library is not in the LIBS: path.

Couldn’t generate AppItem!

Couldn’t create Message-Port for AppItem!

Internal errors.

Can’t read the "gadget"-file...Version of Madhouse

The first line "Madhouse v1.0, BlankerPrefs-Window" is not in the prefs file of this blanker! Perhaps you made a fault in creating a gadget-file,

```
or we made a newer Madhouse which is _that_ new that it is not able to
load the old files (I think I’m not going to change something in the file
structure, so all old gadget-files will be compatible with new ones.)

Wrong Statements in CHUNK:MUI-PREFSORDER.

As the message says, you made a mistake in that chunk.

Couldn’t access the Config-Editor.

Madhouse wasn’t able to start the programm "MadhouseConfigEd". If it was
able to get the settings, it can continue running, and you can change the
ToolType. If the settings were not available, Madhouse has to quit. The
previous requester and this one tell about this situation.
So quit Madhouse (if you need to do so) and change the Tooltype ‘CONFI-
GED’ of Madhouse, so that it includes the complete path and the name of
MadhouseConfigEd. If you do not want to start Madhouse with WBStartup or
with the Workbench, but with the Shell, remember that Madhouse searches
the ConfigEd in the current directory in this case.

The MadhouseConfigEd can only be started by Madhouse itself.

As the requester says, MadhouseConfigEd can only be started by Madhouse,
not by the user. (This was necessary, because both programs use the
Madhouse_Storage-drawer, which can result in conflicts if both programs
run at the same time.)
```

1.16 Programming own blankers

```
Madhouse is an "open system" which allows you to add own modules. The
first question should be:
 Can I use my programming language?

After this, you can write and compile the blanker.
   How should I write my blanker?

Now you can include your blanker into Madhouse’s directory-tree.
         My own directory!

This is all you have to know if you want to write a very simple blanker.
But for a real blanker, you need a BlankerPrefs-window. In fact, you have
to write the "gadget"-file. If you want to open the BlankerPrefs-window,
Madhouse reads this file and creates a "User-Interface" for your blanker.
          The gadget-file

That’s all! If you have problems, please write us!

Now, I want to tell something to people who really want to write a
blanker and to distribute it:
    o Please don’t write a blanker like Lines. Of course it’s your
      decision to distribute a blanker or not. But it would be great if
      the level of the blankers could be kept. Your idea should be a bit
      unusual.
    o Perhaps you think of distributing your blanker together with
      Madhouse. Please don’t do so yourself. This distribution mustn’t be
```

distributed in a modified form.
If you want to see your blanker among ours, please send it to us. Then it will appear in the next version. But in this case it could be that we don’t want to include your blanker (this seems to be arrogant, but what shall we do if someone really wrote a Lines-blanker?!).
Don’t be angry if we return your blanker with some suggestions for improvement...

1.17 Can I use my programming language?

Yes.

Perhaps you think it’s not possible with a language like Basic, but it is. You can use every language which

- offers a compiler to create executable files (I cannot imagine a language which has no compiler, many languages consist of a compiler)
- has commands to open a screen (ok, you will be able to call some library functions, if that’s not the case)
- has commands to read and write files.

Although AMOS can fulfil these requirements, it won’t be a good idea to use AMOS. Aicke wrote his first blankers in AMOS, and now, together with MUI 3.1, they don’t work any more.

1.18 How to write the code.

To answer to this question, I describe what happens when Madhouse starts a blanker.

Every blanker should have settings. You can set them with the Blanker-Prefs window of the blanker. Madhouse writes them to (for example) ".../blankers/BlankerXY/prefs". When Madhouse starts a blanker, it copies the "prefs"-file from the blanker’s directory to the RAM:-Disk ("RAM:-Madhouse_Storage/prefs"). Then it adds two lines: How long the blanker can blank ("Duration") and what the directory for it’s data is. (This information is only necessary if a blanker wants to load data, the "prefs"-file is always in "RAM:Madhouse_Storage/". Such a prefs-file can look like this:

```
3
1
work:tools/Madhouse/blankers/niceblanker
english
```

Here, the blanker has only one setting: 3. This "3" can be changed by the user, perhaps with a slider in the Blanker-Prefs window. Or the "3" means that the user selected the fourth (!) entry of a cyclegadget. How to create the gadgets will be explained later.
The last three lines include always the "Duration"-Time in Minutes (here

```
   1) and the path for other datas. After that, the language preferred by
   the user follows (on Amigas with OS 2.0, it will be always "english"). If
   the blanker has more settings and more gadgets, the prefs-file is

25
$A text, string-gadgets are supported, too!
22
0
work:tools/Madhouse/blankers/niceblanker
english
   longer...

   This blanker has three settings. You see, string-gadgets in the Blan-
   kerPrefs-window are possible, too. But you get the text with a "$" in
   front of it. Here is the duration-value 0, which means that the blanker
   doesn’t have to check when it has to quit, ‘Change Blanker’ was turned
   off.
   After reading this file, the blanker can start. He should open a screen
   and react on the settings. While "blanking", he should test if the user
   pressed a key on the keyboard or clicked with the mouse. Please don’t
   test if the user moved the mouse. This can happen even if the user
   doesn’t want to stop the blanker. If the "Duration"-value is not 0, the
   blanker has to check when it has to stop.
   If the blanker stops because of a user-action, it has to create the file
   "RAM:Madhouse/Stopblank" (by just opening (for writing) and closing it).
   Otherwise, because of the "Duration"-value, the blanker can just quit.
   The screen must be closed, of course...
   Perhaps an error occurres. Then the blanker has to APPEND the error into
   "RAM:Madhouse_Storage/errors".
```

1.19 My own directory.

```
   As you know, Madhouse has an own directory for every blanker. This has
   many advantages, its easy to store all files of a blanker in one
   directory.So create a dubdirectory with the name you want in the blan-
   kers/-directory. Now copy your blanker-executable in it and rename it
   into "blanker". Create an empty file named "prefs" and copy it into your
   sub-directory. (It’s difficult to create a real empty file [lenght=0]
   with a text-editor or such a thing. You can use the prefs-file from the
   developers-drawer.)Then run Madhouse and read the blankers-directory with
   the Path-stringgadget. You will see your blanker in the list. You can
   select/deselect it, but you cannot open the BlankerPrefs-window. You can
   run it by selecting it and deselecting the others, and clicking on "Use".
   Now wait....
   To have settings in your blanker, see the next chapter. Perhaps you won’t
   even try the above example for starting a blanker without "gadget"-file.
```

1.20 The gadget-file

```
   I hope you know already that "gadgets" are the little boxes, where you
   can move the mouse cursor to and click with the left mousebutton...
```

Since Madhouse needs MUI to run (this version and the upcoming ones...), I won’t describe the commands which open a normal Intuition window again. This won’t make any sense, because Madhouse doesn’t look at this part of the gadget-file anyway. (But still you can enter them without getting errors - as you can enter anything in between the chunks.)

Now we want to take a look on such a gadget-file. (Here the gadget-file of Snow.)

```
Madhouse v1.0, BlankerPrefs-Window

CHUNK:LOCALE
english,deutsch

CHUNK:MUI-WINDOWLAYOUT
ColumnGroup(2),
  Label( "Co_llision,Ko_llision" ),
  Cycle( "l", "Windowborders|Screen image,Fensterkanten|Bildschirminhalt" ),
End,

CHUNK:BLANKERINFO
Carsten Jahn
1
1
8000
2
Madhouse
```

You can see:

- The first line includes a text which MUST be in the first line. Madhouse recognizes its gadget-files with this line.
- The gadget-file is organized in Chunks. This means that every Chunk has a Chunk-Header which identifies it (for example: CHUNK:BLANKERINFO). If the Chunk-Header is unknown, Madhouse skips it.
- You mustn’t insert blank lines inside the chunks, but you can have some between two complete chunks. Case is sensitive.

Before I continue explaining every chunk in detail, you may have a short

- overview:The CHUNK:MUI-WINDOWLAYOUT is the most important one, it defines your BlankerPrefs window.
- The CHUNK:BLANKERINFO is not needed for the window itself. The MadhouseConfigEd reads it while reading the whole blankers-directory, and stores the information it Madhouse.prefs, where ‘Madhouse’ can reach it. It contains information about the CPU loading of the blanker, the stack used, etc.
- The CHUNK:LOCALE is for the locale support (used if your OS is 2.1 or better). If it does exist in your gadget-file, you can seperate the languages in CHUNK:MUI-WINDOWLAYOUT by commas.

And now follows the explanation of the chunks in detail. ‘*’ means that the chunk is optional, and doesn’t has to appear in your file.

```
  CHUNK:BLANKERINFO
* CHUNK:LOCALE
```

In the next chapters I explain the usage of two chunks for MUI-windows. As the MUI-syntax differs a lot from the one you know already, I still

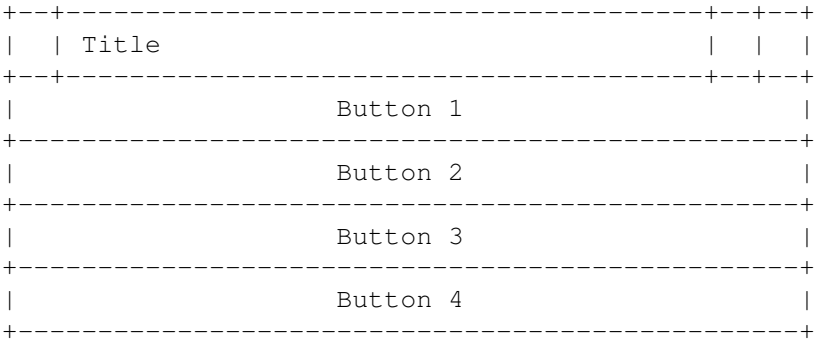
have a lot of work to do... I hope everybody can understand my explanations, although I will write them very fast (I want to FINISH this english doc). If you don’t, please look into the MUI developer kit (available as FD). But I think you will get that. So start the work:

MUI for MUI-Greenhorns
The MUI-Groups
The MUI-Gadgets
The CHUNK:MUI-PREFSORDER

1.21 for MUI-Greenhorns

Creating a MUI-windowlayout is very different. So stop thinking about pixels and fonts, and start thinking about the basic layout of windows.

Here you see a window:



What do you see there? How are the buttons placed? - Vertically. I hope this is the right word, my #?*%-dictionary does not contain the word "untereinander". (If you have a better dictionary, you can look it up :-) But MUI-like we say: The buttons are grouped vertically.

If the window from above was made by MUI, the programmer had just to tell MUI to build a vertical group and give it these four gadgets, and MUI calculated the positions and sizes of the gadgets - very easy.

Perhaps you already know what comes next: horizontal groups containing some gadgets. So we have vertical and horizontal groups and gadgets (of course not just buttons but different types of gadgets, that makes no difference at the moment). But a real window layout is much more complicated. And now comes the point: Groups can contain more than gadgets, they can also contain new groups! It’s the same thing with files and directories: a directory can contain files and directories containing even more files and directories.

< Think a bit. >

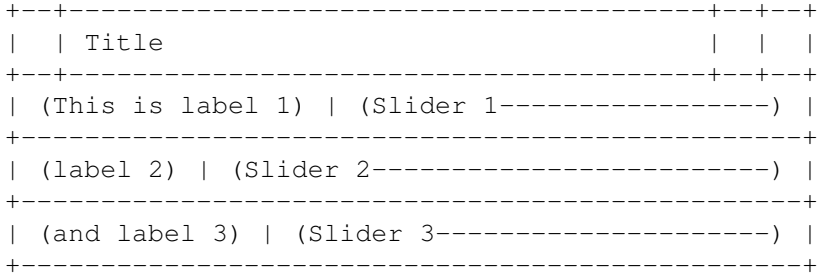
So how looks our window if we replace a horizontal group with two gadgets for button 3? If you think you know the answer, let’s see that you are right: .

But for real windows we need more gadget types. And these other types

have a box containing the gadget itself, like our button, but they also have a text in front of them, like the ‘Exchange Blankers’-Checkmark and all the others have one. We call these texts "labels".

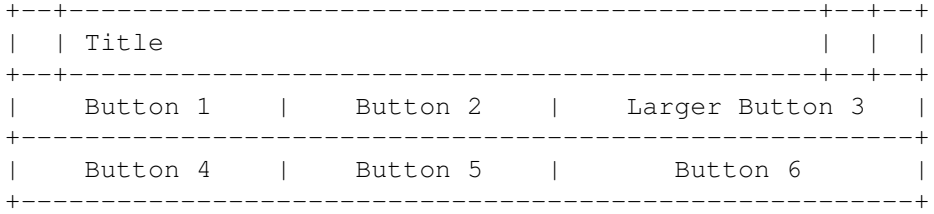
Usually, you define two objects for every gadget: the label and the gadget itself. Madhouse has some short forms of gadget definitions containing also an label on the left side of the gadget, but you use these forms very seldom.

The reason for that is that all MUI-objects (gadgets, labels [and groups]) have some stretching-limitations. You can make a slider as long as you want, but it has a fixed height. This is also true for buttons, stringgadgets and cycle-gadgets. Some objects cannot be stretched at all: labels (containing their text with a fixed width and height) and check-marks. Listviews can be stretched in all directions. So if you use the short form of a slider for three sliders in a vertical group, to build three sliders over another with descriptions on their left sides; you will get these sliders, but the left edges of the sliders won’t match at all. The window will look like that:



(Note that I marked the dimensions of the objects like that: (-----).)

This does not look very nice. This happens because the labels have a fixed width and the sliders are used to fill the box. In this case, you need column groups. Unlike the other groups, column groups have an argument: the amount of columns. So if you create a column group with three columns, containing 6 Buttons, you will get something like that:



Even if the texts inside the buttons have a different lenght, MUI will group them in a way that every part of one column has the same width.

But we need a solution for our problem above, with the sliders. To have the same width for all sliders, we make a column group with two columns, containing a label, a slider, a label, a slider, a label and a slider (in that order). Then we get

```
          |  | Title                                    |  |  |
          +--+------------------------------------------+--+--+
          | (This is label 1) | (Slider 1----------------) |
          +-------------------------------------------------+
          | (        label 2) | (Slider 2----------------) |
          +-------------------------------------------------+
          | (    and label 3) | (Slider 3----------------) |
          +-------------------------------------------------+
```

That’s much better! But how does this look like in the gadget-file? First, I will tell you how our first example looks like:

```
CHUNK:MUI-WINDOWLAYOUT
VGroup,
   Button1,
   Button2,
   Button3,
   Button4,
End,
```

In our new chunk "MUI-WINDOWLAYOUT" we put a vertical group. This group reaches up to "End". It’s the same with IF and ENDIF in a programming language: every ENDIF matches to one IF. Inside the group, we write the objects which are assovciated with it, in this case four buttons. Every line ends with a "," - but you are not allowed to merge two lines to one! You don’t have to "indent" the lines like I did, but you should do so because it’s easier to read. You can use spaces or TAB’s for that.

And how about our column group from above? For didactical purposes, I use a cycle-gadget instead of the third slider.

```
CHUNK:MUI-WINDOWLAYOUT
ColumnGroup( 2 ),
   Label( "_Label 1" ),
   Slider( "l", 1, 10 ),
   Label( "L_abel 2" ),
   Slider( "a", -5, 20 ),
   Label( "La_bel 3" ),
   Cycle( "b", "First entry|Second entry|Third entry" ),
End,
```

Now the chunk contains a column group, as we need one for the correct formatting of the labels. The "(2)" says the column group to make two columns. A column group has to be filled up to the right-most column. In this case, the group can have 2, 4, 6, 8, ... "children". Child is the MUI-word for an associated object. If a column group as five columns, it can respectively have 5, 10, 15 and so on children. If you don’t want to fill every place in a column group, there is a "dummy" object for that, which I will explain later.

As the last example is one which could be written into a gadget-file without changes, you can see there the definitions for labels, sliders and cycle-gadgets. A label is done by the keyword "Label", including the text of the label in this form: Label("Text"),. You can use an underscore _ to underline the following char. This is used to show the user which key of the keyboard can be pressed to control the gadget.
A slider comes next: this type of object needs three arguments: the control-key (which should be marked in the label before), the minimum and

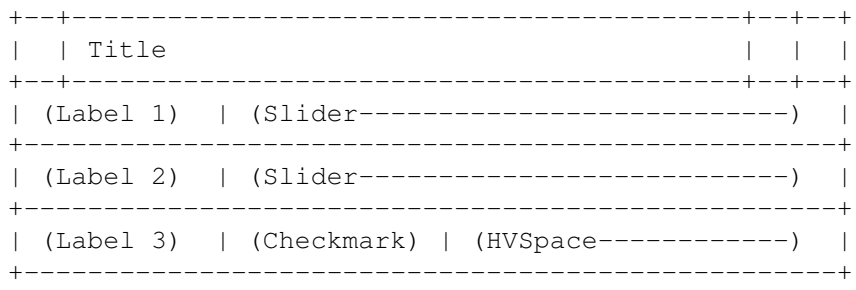
the maximum value. So the first example produces a slider which ranges from 1 to 10. You can also use negative values, see the second slider. The cycle-object needs two parameters: the control key, which is always the first parameter of every gadget, and the different cycle-entries, separated with "|". On my german keyboard, I can find this char next to the backspace-key.

Please note that you cannot use every control-key you want: you can only use non-capital letters and you cannot use the chars o, t and c, because they are already used by the three buttons on the bottom of every BlankerPrefs-window.
And now comes an example where we need to put one group into another. Please wait patiently while I’m explaining the situation where we need one... (just building up the problem for my solution :-/)

Perhaps (oh, I’m sure...) you have noticed that MUI arranges the gadgets automatically (that is why we don’t need pixel-coordinates) and dynamically. This means that almost every MUI window has a size-gadget, which allows us to size it, and MUI recalculates the gadget-positions so that everything fits into the window again. But have you noticed that you can size a lot of MUI windows only horizontally, the height is fixed? (Try some BlankerPrefs-windows). Can you imagine why this happens? Every MUI-object (gadgets and groups, but we’ll start with gadgets) has its minimums and maximums for width and height. Sliders, string-gadgets cycle-gadgets and some more have a fixed height (minimum height = maximum height), but no fixed width (maximum width is VERY big). To scale the height of a cycle-gadget would make the cycle-gadget looking silly. Lists have neither a fixed height nor a fixed width, checkmarks have both. So we have answered an half of our question: you can size a lot of MUI windows only horizontally, because a lot of windows contain objects with a fixed height.

But we still need the problem. Here it comes. Try to replace the cycle-gadget by a checkmark (see our last example, slider - slider - cycle). Now, the result looks quite strange: as the two sliders and the checkmark are placed in the same column, MUI wants to give them the same width. As MUI fails in scaling the checkmark horizontally, it gives the sliders the width of the checkmark, which is of course too small for a slider. The window has no size-gadget at all, because the group inside the window cannot be sized (the left column is blocked by the labels, the right by the checkmark, both objects with fixed width). We have to help MUI, and make the checkmark bigger. (This is ... impossible.) But we could replace the checkmark by a horizontal group, containing a checkmark and a slider. Now, the window could be sized horizontally again, because the column group can be sized again, because it has a column (the right one) which can be sized, because all the members in this column can be sized, because even the horizontal group (containing a checkmark and a slider) can be sized, because this group has one object (the slider) that can be sized. Uff... "can be sized" means here "can be sized horizontally", MUI does the same thing for vertical sizing.

- Hey, my window looks stupid with all these sliders which I filled in after I’ve heard your explanation! - No problem, MUI offers of course another solution for our problem. It has an object that can be sized to unlimited dimensions and which is... invisible. We call it HVSpace. Our little ANSI-graphic - AmigaGuide has no drawing tools :-(is here:



And you code this like that:

```
CHUNK:MUI-WINDOWLAYOUT
ColumnGroup(2),
   Label( "_Label 1" ),
   Slider( "s", 1, 10 ),
   Label( "L_abel 2" ),
   Slider( "a", -5, 20 ),
   Label( "La_bel 3" ),
   HGroup,
      CheckMark( "b" ),
      HVSpace,
   End,
End,
```

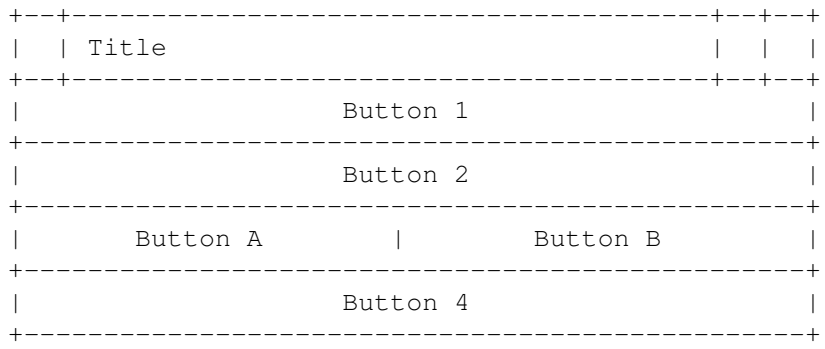
I hope you got it now. The HVSpace is also usefull in a second case: if you have a big columngroup and do not want to put objects in all cells, you can assign the HVSpace to the cell which should be empty.

To get further impressions of MUI-layout, you can look into the gadget files. This was not the whole MUI documentation for Madhouse, just the basic knowledge. All types of gadgets and another feature of the groups can be found in the next chapters.

If you have understood the basics of MUI, you can also write your own MUI application with your knowledge. The code differs a bit (since you are programming in a real language, and don’t write instructions into a small file for Madhouse). Then, a big compiler goes through your code and not my self-made MadhouseConfigEd. Where is the difference?! A compiler is a program which some people work on, for years. My interpreter for the MadhouseConfigEd was done by myself in four days. Please excuse me,

- I have not included every MUI-feature into the interpreter. You cannot use paged groups, the weight of every object is fixed to 100, you cannot use CustomClasses (why should you). But you can do a lot. And the MUI BlankerPrefs windows generated out of the gadget files look as good as ’real’ MUI applications.
- I was too lazy to handle ANY error in the gadget-file. You can make thousands of errors, and Madhouse does not crash (I hope), but you do not get an indiviadual error message (or you get no one at all) for every mistake. You will recognize if something is wrong (the window will look strange), and it should not be difficult to find the error in such a small file.

1.22 The solution



1.23 The groups

Of course I don’t explain the basics of MUI-groups again (see greenhorn chapter).

HGroup and HGroup("Grouptitle")

The HGroup (which places the objects inside horizontally) can have a frame around it and gets visible. The grouptitle will be shown in or above the frame. If you want to have a frame, use the syntax above.

You can do the same thing with VGroups.

ColumnGroup(x) and ColumnGroup("Grouptitle", x)

x stands for the amount of columns, if you use the second syntax you can have a grouptitle.

1.24 The gadgets

Some standards in this chapter

- Key contains a non-captial letter which is used as the control key for the gadget. The user can control the gadget by pressing this key on the keyboard. Example: Slider("a", 5, 10) makes a slider which can be controlled with "a". You cannot use the chars "o", "c" and "t", because they are already used by the gadgets on the bottom of every BlankerPrefs window.
- Text contains a text which will be displayed on the left side of the gadget. Example: LabelCycle("Color _selection", "s", "Red|Green|Blue") would draw "Color selection" next to the gadget, with the char "s" underlined. This is used to show the user the control key of the gadget.

Slider(Key, Minimum, Maximum)

creates a slider. It ranges from Minimum to Maximum.

LabelSlider(Text, Key, Minimum, Maximum)

like above, but has an additional label (text) on the left side.

CheckMark(Key)

creates a Checkmark.

Cycle(Key, Entries)

creates a cycle gadget with some entries. "Entries" has to contain all entries of the cycle gadget, separated by "|". Example: "RGB|HSV|CMYK". MUI gives the first entry the number 0, so if you select "RGB", Madhouse will write "0" into the prefs file (the same thing with normal gadgets).

LabelCycle(Text, Key, Entries)

like above, but has an additional label (text) on the left side.

String(Key, Length)

creates a string gadget which can contain max. ‘Length’ chars. The upper limit is 500 chars.

LabelString(Text, Key, Length)

like above, but has an additional label (text) on the left side.

Label(Text)

This object gives you the posibility to place your objects better. If you separate gadget and label (as you do it normally), the windows will look much better. Use this object and a gadget without Label... in front of its name.

LLabel(Text)

like above, but this object aligns the text left, not right. This is useful to place another text on the right side of a slider (e.g. "seconds", "minutes", "m", "Objects", things like that). An example of this object can be found in the gadget file of Waves.

HVSpace

The important dummy object, its usage was explained in the greenhorn chapter.

HBar

An elegant object which draws a horizontal bar to separate some gadgets. Should be used only in horizontal groups.

VBar

like above, but draws a vertical bar. An example can be found in the

```
   gadget file of Stars.
```

1.25 The CHUNK:MUI-PREFSORDER

```
   "How can MUI get along with only one chunk?!" - It can’t! In some rare
   cases, you will need the Chunk MUI-PREFSORDER.

   But first a problem to explain its usage:
   Madhouse writes the settings of the BlankerPrefs window in that order, in
   which the gadgets were defined. BUT: You can define normal gadgets in the
   order you want, you cannot do this with MUI gadgets. You have to define
   them in an order that makes the window look good.
   With this Chunk, you can easily re-order the settings which are written
   into the prefs file. First, you give every gadget a name:

CHUNK:MUI-WINDOWLAYOUT
VGroup,
   LLabel( "An extract of Stars" ),
   ColumnGroup( 2 ),
      Label( "_Maximum" ),
      Maximum = Slider( "m", 1, 8 ),
      Label( "M_inimum" ),
      Minimum = Slider( "i", -7, 1 ),
      Label( "C_anges" ),
      Changes = Slider( "a", 0, 15 ),
      Label( "_Start" ),
      Start = Slider( "s", -7, 8 ),
   End,
End,

   Without MUI-PREFSORDER and the names, your prefs file would look like

Setting of Maximum
Setting of Minimum
Setting of Changes
Setting of Start
Duration in minutes
Path to the directory of your blanker
Language

   that:And now comes the point: if you use the names and write this into
   your gadget file

CHUNK:MUI-PREFSORDER
Changes, Start, Minimum, Maximum

   you get this prefs file:

Setting of Changes
Setting of Start
Setting of Minimum
Setting of Maximum
Duration in minutes
Path to the directory of your blanker
Language
```

Ok, you can live without this Chunk, but sometimes it is helpful.

1.26 The CHUNK:BLANKERINFO

WARNING: Madhouse reads the data in this chunk while reading the Blankers-directory, NOT (!) while opening the BlankerPrefs-window. If you change something in this chunk, Madhouse will use the old cpu-loading and stack-values until you use the Path-gadget!

```
CHUNK:BLANKERINFO
name
version
cpu-usage
stack
WBDisplay
Protocol
Category
```

- name = Your Name.
- version = The version of your blanker.
- cpu-usage = This value can be 0 or 1. You should set it to ‘0’, if your blanker needs none (or very little) CPU-performance, otherwise ‘1’. Madhouse needs this information to check easily if it can run your blanker while a raytracing program is calculating, for example.
- stack = The stack depends on your compiler and on your program. Try first 5000. If your blanker crashes or aborts, try more... (See the documatation for your compiler). Don’t use values that cannot be devided by four.
- WBDisplay can be 1 or 2. 2 means that your blanker copies the Workbench or the frontmost screen onto his own screen. 1 is right if you never do so. (Madhouse needs this information to know if it occasionally has to undimm and close it’s tiny black screen before starting your blanker; see Blankers-page, ‘Shows WB’.)
- Protocol has to be ‘Madhouse’.
- Category: If you want, you can categorize your blanker. At the moment, this setting does only influence the sorting of your blanker in the list. One of these keywords can be used in this line:
    - Anim, for blankers which use small, animated pictures (like FlyingToasters);
    - Clock, if your blanker has something to do with date and time;
    - Pyro, for all fireworks and fountains;
    - Text, if your blanker displays some text;
    - Fract, for effects which come out of fractals;
    - View, for blankers which display whole anims and pictures (in contrary to Anim!)
    - Algo, for algorhythmic blankers. Every program can only consist of algorhythms, so this category means cellular automations like Life;
    - Sim, for simulations of rain or software failures and so on;

- 3D, if your blankers draws 3D objects or moves them in a threedimensional way;
- Pixel, if the effect does consist of single, isolated pixels, and it is no fireworks ;-)
- Cycle, for effects which would look very boring without color cycling;
- WB, for all blankers which copy the frontmost or the Workbench screen and do their work on it;
- Geo, if you are drawing geometrical objects (circles, rectangles, lines in special forms), this is the right category for you;
- Lines+Splines, if your blanker draws these well-known simple lines or splines;
- ?, if you don’t know at all, or if no category matches the character of your blanker.

1.27 The CHUNK:LOCALE

```
CHUNK:LOCALE
language1,language2,...
```

While localizing (giving several languages to a program) Madhouse, I found out that I need to localize the BlankerPrefs windows, too. Therefore, the chunk MUI-WINDOWLAYOUT needs the gadget texts for every language, not just for one. Example:

```
CHUNK:MUI-WINDOWLAYOUT
ColumnGroup(2),
   Label( "_Mode,_Modus" ),
   Cycle( "m", "Bouncing Point|Wusel|Random,Springender Punkt|Wusel|Zufall" ),
   Label( "Wusel _lenght,Wusel_länge" ),
   Slider( "l", 1, 20 ),
   Label("_Sound,To_neffekt"),
   HGroup,
      CheckMark( "s,n" ),
      HVSpace,
   End,
End,
```

The Locale-chunk concerns all textdefinitions (recognizable with the "-char) in the gadget file. As you can see, every text has a ,-char now. The comma seperates the different languages. The left parts of the texts are the english texts (1st language), the right parts the german ones (2nd language). Other languages could follow. So the right CHUNK:LOCALE looks like this:

```
CHUNK:LOCALE
english,deutsch
```

Madhouse works like this:

- If the MadhouseConfigEd recognizes on startup, that this is no OS 2.1 or higher Amiga, it uses english as the default language. Otherwise it uses the language set with the Locale-editor (from the Workbench), for example deutsch.
- Before opening a BlankerPrefs window, MadhouseConfigEd searches for

the CHUNK:LOCALE. If there is none (this Chunk is optional!), it prints the texts into the BlankerPrefs window without any change. But if there is one, MadhouseConfigEd looks into the line with the languages and searches the user-defined language ("english" if this is an OS 2.0 Amiga). If it can find this language, it memorizes the amount of commas skipped, and will skip this amount of commas for every text definition. If it cannot find the language, it uses the first one (SHOULD BE "english"). If some text in your MUI-WINDOWLAYOUT has no comma at all, the text will be printed as it is.

As you can see above, in cyclegadget defintions the comma has a higher priority than the |-separator. Madhouse does the locale-parsing first, and then separates the cycle-entries. The control-keys are texts, too. So you can define different keyboard "maps" for every language, perhaps "_Sound,To_neffekt" and "s,n". If a translation produces the same word ("_Level,_Level" and "l,l"), you don’t have to use the locale-feature and can write "_Level" and "l". Madhouse does no locale-parsing if it cannot find a comma in the string.

Use non-capital letters for the language names, and use the language itself to write the name of it. Examples: français, english, deutsch, ...

I hope I made all things clear - ..

1.28 about MUI

This application uses

MUI - MagicUserInterface

(c) Copyright 1993/94 by Stefan Stuntz

MUI is a system to generate and maintain graphical user interfaces. With the aid of a preferences program, the user of an application has the ability to customize the outfit according to his personal taste.

MUI is distributed as shareware. To obtain a complete package containing lots of examples and more information about registration please look for a file called "muiXXusr.lha" (XX means the latest version number) on your local bulletin boards or on public domain disks.

If you want to register directly, feel free to send

DM 30.- or US$ 20.-

to

Stefan Stuntz
Eduard-Spranger-Straße 7
80935 München
GERMANY

1.29 All our Blankers

Here, you can look up the functions and settings of the blankers. Unfortunately, the AMOS-Blankers made troubles with MUI 3.1, and are not included in this release. Even with MUI 3.1, they can be used with some limitations, so you can find them in Aminet with the filename ‘Mad-AMOS.lha’ in util/blank. Look out for new blankers, ‘Mad#?.lha’-files will appear soon (I hope...).

BouncingPixel
Fireworks
Flow
FlyingToasters
Glitter
Nautic
Shuffle
Snow
Soccer
Stars
Waves
Wusel

1.30 BouncingPixel

BouncingPixel

By Aicke Schulz, version 1.1

Aicke rewrote this blanker in AmigaE, it was part of the old AMOS-CrazyPixel blanker.
--- No Options ---

CPU-Loading: low.

1.31 Fireworks

Fireworks

By Carsten Jahn, version 1.

This blanker shows a nice and colorful firework on your screen. It has four effects. You can use the sliders to set how often one effect should appear. If you don’t use too high values, FireWorks is able to change the colors while one register is unused.

- Spirals sets how often Spirals are used.
- Jets sets how often Jets are used.
- Balls sets how often Balls are used.

```
    o Color-Jets sets how often Color-Jets are used.
    o Pixelspeed: Perhaps you've noticed that this effect becomes slower
      and faster all the time. Try different values with the Pixelspeed-
      slider to get the right timing for your machine.

CPU-Loading: high.
```

1.32 Flow

```
Flow

By Aicke Schulz, version 1.

Flow draws an interference pattern and lets it flow with color cycling.
After a while, Flow will draw another picture.

    o Cyclespeed Flow can flow fast and slow ;-)
    o Cycledirection Flow can flow in two directions, guess which.
      'Random' is possible, too.
    o Wait Flow can wait between two pictures. This is the time in
      seconds.
    o Colors Some palettes are available. Select 'Random', and Flow will
      choose one itself.

CPU-Loading: high.
```

1.33 FlyingToasters

```
FlyingToasters

By Carsten Jahn, version 1.

Yeah, they are back. Being heavily inspirated from the AfterDark-Toasters
(not the SuperDark-Toasters), I painted the graphics with PersonalPaint.
They need a 16-color-hires-interlaced-screen and have 6 anim-phases.
Perhaps I'll add a check for the "DblPal"-modus, but I think graphics
don't flicker too much.

    o Toasters sets the amount of toasters on the screen.
    o Toasts sets the amount of toasts on the screen. The toasters will
      "overdrive" them, but not crash them.
    o Speed can be a value between 1 to 7. 7 is the fastest mode.
    o Jam adds jam to the toasts. The perfect breakfast!
    o Double Buffering enables, as the name says, double buffering.
      Normally, the blanker draws only on one screen, and you can perhaps
      see the "Blitter" working (depends on your Amiga and the settings).
      With double buffering, the program will draw on one screen (which
      you can't see) and shows another, older screen. Then it will swap
      them. Double buffering needs more memory. If it is not available
      the Toasters won't use double buffering.
    o In/Out Moving With this option switched off, the Blanker will put
      and erase the Toasters immediately on the program's start / end. If
```

you use "In/Out Moving", the Toasters will flew into the empty screen on startup. Before the Durationtime is over, the Blanker won’t put any more Toasts, so that the screen will be left empty again. The last thing is only possible if "Change Blanker" was checked.

CPU-Loading: High.

1.34 Glitter

Glitter

By Aicke Schulz, version 1.1

It’s not the last program-technical invention, but a nice effect that shows ... shows what? Glittering stars.

- Number sets the number of stars. This will slow down the startupprocess, but not the animation.
- Speed sets the speed of the "glittering".

CPU-Loading: Low.

1.35 Nautic

Nautic

By Carsten Jahn, Version 1.

I’ve seen something like that in a demo, and programmed it myself. Because of a drawing trick, this blanker needs an AGA-Amiga or a graphic card.

- Fast Switches in the slightly faster mode.
- Wait How many 1/5secs. wait time until Nautic erases the picture and draws a new one.
- Number of edges sets the amount of edges for the drawing template. ‘4’ is a nice setting.
- Random The amount of edges changes with every new figure, no matter what you set.
- Colors Which combination of colors you prefer.
- Monochrome The first two color combinations can be used with small color ‘mistakes’, which can look nice. If you set the checkmark, these mistakes won’t be used.

CPU-Loading: Low.

1.36 Shuffle

Shuffle

By Carsten Jahn, version 1.

Have you ever seen a modulary screen-blanker without a "Shuffle"-module? Here is our one. But I didn’t wanted to write such a simple Shuffle, but one with some new features.
So this Shuffle is able to restore a puzzle.
It is AGA-Chipset-sensitive and recognizes other Screenmodes like DblPal and others. You can use more colors, too. It will turn off the 3D-Grid if the colors are "wrong". And I hope it will run on graphic cards which use Screenmodes (tell me about it, I don’t have such a card).

- Mode sets the Shuffle-mode. "Shuffle" is the normal Shuffle as you surely know it. The other modes can only be used with "Change Blanker" (main-window) activated, because the blanker has to know when to stop. "Shuffle & Restore" first shuffles the screen, and then solves the puzzle. "Restore" starts with a shuffled screen and then restores it in the "Duration"-time.
- Speed sets the speed of the shuffling.
- Grid turns the 3D-Grid on or off.
- X/Y number sets the size of the tiles.

Please don’t be confused if it takes a while before the blanker starts, if you select a high ‘Duration’ time and the mode ‘Restore’. Shuffle has to prepare the ‘Restoration’, you know...

Errors: perhaps a text like ‘The "Shuffle & Restore" mode and the "Restore" mode can be only used if "Exchange Blanker" is activated.’ will occur. This means that you tried to use a Restore-mode although the blanker cannot know when to finish, but he has to know that because he does not know what to do after the restore is finished.
CPU-Loading: Low.

1.37 Soccer

Soccer

By Carsten Jahn, a lot of anims by Aicke Schulz, version 1.
Hey, have you ever seen something like that? A soccer game as a screen saver. Forget your TV-set, now you can watch soccer while setting in front of your Amiga. Perhaps, with this blanker, soccer will become more popular in the U.S.A.
Soccer has no sound, it’s your task to cheer.
If there is enough memory, Soccer uses Double Buffering.

- Line-up White/Red With these switches you can select the line-ups for the two teams.
- Keeper intelligence Of course, there is no intelligence in your Amiga, but this slider allows you to set the "artificial" intelligence of the goalkeepers.

CPU-Loading: High.

1.38 Snow

Snow

By Carsten Jahn, version 1

Let it snow on your workbench!

- Collision You can have collision with the window-borders (the best thing for normal screens, the snow will lay on the windows soon!) or with the contents of the bitmaps (Screen image).

CPU-Loading: high.

1.39 Stars

Stars

By Carsten Jahn, version 3

What should I say... This is the Madhouse-version of the well-known starblankers. They have an new effect that I haven’t yet seen on an Amiga: The turns and the backwards gear. Try it out. Many settings.... If Stars just draws bizarre things on your screen, you’ve installed the wrong version (for Amigas without graphic extension).

- Number of Stars sets the number of stars.
- Normal Movements (movements ON the z-axis)
  - Maximum speed of the stars.
  - Manimum speed of the stars. Negative values: backwards gear.
  - Changes sets how often the stars change their speed (between the two values you can select with Minimum...Maximum.
  - Start sets the start speed.
- Turns (movements AROUND the z-axis)
  - Maximum speed of the turn.
  - Acceleration sets how fast the stars get their new speed.
  - Changes sets how often the stars change their turn-speed.
  - Start sets the start speed.
- Shifts (movements on the x- and y-axis)
  - Speed sets the speed of the shiftings.
  - Usage How often this effect will be used. "0" means never, "10" means always.
  - X/Y Which axis (or both?) will be shifted.

CPU-Loading: high.

1.40 Waves

Waves

By Carsten Jahn, a lot of palettes by Aicke Schulz; version 1.

Here is a blanker which mades very interesting Plasma-effects. I don’t
want to explain in detail how this was made (because I had even problems
explaining it in german, sorry, but you perhaps know already).
On the left side of the BlankerPrefs-Window you see a selection of
color-palettes. Waves uses one of these which are activated (checkmarks).
If you deactivate all colorsets, Waves will choose itself.
On the right side are the parameters which influence the shape of the
waves:

- o WaveSize influences the thickness of the waves.
- o XWaves sets the width of the waves.
- o YWaves sets the height of the waves.
- o XSize is the maximum of the horizontal sinus-arc.
- o YSize is the maximum of the vertical sinus-arc.
- o XSpeed sets the speed of the horizontal screen-scrolling.
- o YSpeed sets the speed of the vertical screen-scrolling.

It is useful to experiment with parameters.

CPU-Loading: Low.

1.41 Wusel

Wusel

By Aicke Schulz, version 1.1

As BouncingPixel, Wusel is from the old CrazyPixel-blanker, which was
originally written in AMOS.

- o Length Length of Wusel.
- o Speed Use this to set the speed of Wusel.

CPU-Loading: low.

1.42 The Authors

The Authors

Hello,
we are the authors. Madhouse is Shareware. If you use Madhouse regularly,
you are asked to send us DM 20 or US$ 15, together with the registration
form. We will return a keyfile to you, what you should place in your
S:-Directory. With this keyfile,

- o Madhouse won’t bother you with ‘Nervscreens’ and
- o Madhouse will run Garshne- and Swazblankers. (Otherwise you can run and configure them in MadhouseConfigEd only.)

Here are the addresses:

Aicke Schulz
Neudeckerweg 118

D-12355 Berlin
Germany
Telephone: 030 (Berlin) / 664 48 22

After March ’96:
Carsten Jahn
Kuckucksruf 34
D-16761 Stolpe-Süd
Germany