# Haute-Cuisine Menu Editor (MenuEdit)

| COLLABORATORS | | | |
|---|---|---|---|
| | *TITLE* :<br><br>Haute-Cuisine Menu Editor (MenuEdit) | | |
| *ACTION* | *NAME* | *DATE* | *SIGNATURE* |
| WRITTEN BY | | January 2, 2023 | |

| REVISION HISTORY | | | |
|---|---|---|---|
| NUMBER | DATE | DESCRIPTION | NAME |
| | | | |

# Contents

# Chapter 1

# Haute-Cuisine Menu Editor (MenuEdit)

## 1.1 ME Documentation (13.10.1997)

Documentation for

The HCMenu Editor
Version 1.1





## 1.2 What is the HCMenu editor all about?

Before answering this question it's better to answer this question:
"What is the Haute-Cuisine Menu?"
Simplysaid: The HCMenu is a programm which turns a simple textfile into
a menu. This menu is a fully cofigurable window containing a number of
items. Each item may be selected (activated) by a simple mousclick.

As already stated, the menu works with a textfile. With the help of a
standard texteditor like "Cygnus Ed" or the terrifying "Ed" from the
workbech disk is is possible to add items etc. to a menu or make a
complete new menu.
That is not hard to do, but there are more simpler ways to maintain
a menu. And that is the reason I coded the HCMenuEditor. It's a tool
which enables the user to maintain his menus with the mouse, since
that is what the Amiga makes great...

## 1.3 Copyrights and disclaimer stuff.

Both the HCMenuEditor and the documents may be free copied as long as
they stay in THE ORIGNIAL STATE. All copyrights belong to the author.
I did not make the HCMenu nor the HCMenuEditor to make a profit. I think
the growth of the Amiga is also made possible by the quality and quantity
of th public domain software.
One thing that also relates to the price is that I cannot be held
responsible for any problems which could occure. The use of this software
is completely at YOUR OWN RISK!

Although I don't want any payment for this software, I realy like it
when people who use it send me a postcard with some new ideas, or some
bugs. To me the HCMenuEditor is more worth than it's price (nothing)...

For problems, bugs, thanks or loveletters contact me through phisical
mail, writing to:

E-Mail: eto@icns.nl
WWW: http://www.icns.nl/users/eto

E. Th. van den Oosterkamp
Vijverstraat 11
4103 XX Culemborg
Netherlands

## 1.4 Some thanks to....

Sofware cannot exist without it's beta-testers, testsoftware or people
sending in ideas. Thats why I thank here some people who helped me in
one or another way during the devellopment of the Editor.

M de Reuver - Thanks for some (verry good) suggestions in the matter of
userfriendlyness and the definition of functions.

M van Aalten - Thanks for the bug in rev 77! *;-)

M Blomsma - Your suggestions were helpful indeed. They made functions a lot user-friendlier.

T.F.A. - Thanks for the upgrading of Asm-One! After the release of version 1.25 there is no alternative! Asm-One rules!

Michael Sinz - Thanks for upgrading the Enforcer! It helped me to remove some nasty things which I would never have found without the help of Enforcer!

Now I want to ask (the few?) user to write me when you've got any suggestions or bugs. Things I do not know of I cannot change! Look for

my address
somwhere else in this guide!

Please try to give as much of information when making a bugreport. Things like revisionnumbers and used functions are vital!

## 1.5 Which system-demands do we've got?

The Editor only works with an operating system version 2.04 or higher. There are no demands for the amount of free memory. As long as there are a 100kBytes free, the editor should work.

The editor needs these system-files:

- Libs:AmigaGuide.library
- Libs:PowerPacker.library
- Libs:Asl.library (*)
- Libs:DiskFont.Library (*)

(*) These files *must* exist! When one of these are missing will the editor stop using a well fitting errormessage (missing....) :(

When the AmigaGuide.library is missing the possibility of on-line help will be disabled. For other functions this libraty is not needed. The powerpacker.library is used to compress and to decompress the menu-files. When the library is missing the editor will work properly, but will refuse to decompess or compress menufiles. The powerpacker.library is made by Nico François and is public domain.

The memoryconsumption depends on the size of the menuwindow and the amount of items in it. The editor does not need MegaBytes of memory. Will you although receive a message since there is a lack of memory, please *DO* save your menu using a new name, since some items could be lost.

## 1.6 Differences comapred to the previous versions.

```
                V1.0 -> V1.1
============

New functions:
- Contents of the ClipBoard will be stored on disk between sessions.
- The ClipBoard now has its own user interface: the
                ClipBoard Window
                .
- The
                ItemEditor
                 is very improved (since I say so).
- It is possible to
                import
                 some other menu formats.
- The length of the "action" is not restricted anymore.
- The editor is now completely fontsensitive.

Removed Bugs:
- Menu was always sorted alphabetically, even if the user wanted something
  else. This is history now.
- A menu ending with an "Invisible" item refused to load.
- When a menu contained a "normal" item without a name (the way the are
  produced by the "Insert" function..) it was written incorrectly.
```

## 1.7 Quick-start for professionals.

```
                In the next section some information meant for the more  ↩
                   experienced
users of the Amiga operatingsystem. Things meantioned here can (when
important enough) also been found at other places in this guide. There
they'll be better explained. Reading this section is NOT neccessary
for using the Editor...



Starting the editor
===================

The editor may be started from both CLI/Shell or the WorkBench. When
started from CLI/Shell the editor uses a CLI-Detach routine to free
the cli, "Run" is not needed.
The menu that will be loadad during the startup of the editor is the
same menu that is active in the HCMenu (when there is a HCMenu running).
When there is no HCMenu running, the mainmenu will be loaded. The name
of the mainmenu is loaded from the HCMenu-snapshot file.
If this snapshot-file also misses, the file S:Main.Menu is loaded. If
that also misses, nothing will be loaded...



Preference files
================

There are three types of preference files. All of them are loaded during
```

the menu-startup. When a file misses the default settings will be used.
Now follow the files, their contents and some other information:

ENVARC:HCMenu.prefs
---
This file is used by both the HCMenu and the editor. This file is
called the "general snapshot" or sometimes the HCMenu-"preferences".
This file contains the information for the HCMenu: it's screenposition,
it's size, the used font, number of items/line, usage of borders etc.
Using the editors
Pref
Window some of these things
are also alternable.
Saving of this file is only possible with the prefwindow.

ENVARC:HcMe.Prefs
---
This file is only used by the editor. It is called the "settings" file.
In this file all settings and preferences of the editor are stored. This
is information like: position of all windows excluding the menuwindw
(since those are in the snapshot) this also includes all ASL-requesters.
Also filenames, paths. patterns, etc are stored in this file.
This file may be saved using the
settings
Window.

<MenuFile>.snap
---
It is possible to give each menu it's own (local) snapshot. In this
snapshot settings like the windowposition -and size and the number of
items per line are stored. The name of the file is the same as the name
of the menu to which it belongs, but it extended with ".snap".
When the menu is written to disk, this local snapshotfile is written
with it. In the
settingswindow
the writeing of the local snapshot may
be turned off with
Update Snap
.

On-Line help
==

For the on-line help the editor uses the famous AmigaGuide system. Nor-
mally it is sufficient to put the "Me.Guide" file in the same directory as
the editor. When there are still problems using the on-line help this
may help:
- Make a directory where you want to store the guides of all
software you have (e.g.: "SYS:Guides").
- Fill the directory with the guidefiles (including "Me.Guide").
- Use the "SetEnv" DOS command to fill the "AmigaGuide/Path" var-
iable with the path to the directory you just made.
(Example: "SetEnv ENVARC:AmigaGuide/Path SYS:Guides")

The On-Line help SHOULD work now... ;^)

## 1.8 The on-line help.

These are the sections which are available for on-line help during the
execution of the editor (when the installation is O.K.).

The Buildwindow
- Automatically generation of a menu.

Het Clipboard
- Using the vlipboard.

The ItemEditor
- Edit the name and function of a menuitem.

The MenuBar
- All functions found on the menubar.

The PrefWindow
- The "Look" of the menu in your hands.

The Settings
- Setting the editor to one's taste...

The ToolWindow
- The most used functions in one hand.

## 1.9 The selection of items.

By clicking items in the menu these can be selected. When an item is
selected this can noticed by the colorchange. By clicking the item again
it will be deselected.

By selecting multiple items it is possible to perform the same action
on more items. In this way, the user is able to delete for instance more
items at a time.

## 1.10 Using the ClipBoard.

The ClipBoard is the place where the items reside between the sub-
sequential
Copy
,
Cut
en
Paste
actions from the
ToolWindow
.

At the left side of the window a list is displayed. This list contains
the names of the items that are on the clipboard. The topmost item in
the list is the item that will be used for the next comming Paste.
When items are added to the clipboard (using the Cut or Copy functions)
they will be add at the top of the list.

The clipboard window may be closed with the help of the close gadget. This
will only close the window as the clipboard itself is always present.
When the window is closed the clipboard will still function normally.

Top
---
With the "Top" gadget the clipboard-item that was selected in the list
will be transferred to the top of the list. This will make the item the
first item to be used during the next Paste.

Edit
----
Open thee
ItemEditor
for the items selected in the clipboard list.
Double-clicking an item will also open the ItemEditor.

Delete
------
remove an item from the clipboard. There is no way to get it back!

Clear
-----
Clear the complete clipboard. There is no way to get the items back!

## 1.11 The Item Editor

With the ItemEditor the items in the menu can be edited. When ↩
multiple
items are selected at the moment the itemeditor opens, each item will be
edited separately. It is not possible to edit a number of items at once.

The Editors main controls:

```
                ItemName
                 - The name of the item.

                ItemAction
                 - The commands for the item.

                ItemType
                 - The type of the item.
```

The arrow buttons at the left of the window will move a selected command upwards or downwards in the list.
The "Del" button between the arrows will remove the selected command from the list.
the checkboxes are part of the ItemAction.

The editor may be closed with the "Okay" button. Any change made will be visible in the menu directly. Closing the editor with the "Cancel" button will discard any change made.
The "Help" button will open this text.

## 1.12 Changing the item name.

The "ItemName" stringgadget is the upper most gadget in the window. In this gadget the name of the item is specified. A name of 80 characters is the maximum size.

It is not possible to leave an item unnamed. Only "invisible" items have no names. When the name is not filled in a requester will warn.

The name of the item is of no importance to the menueditor or the hcmenu itself. Offcourse it is for your own good to select short and good item names. One name may be used multiple times: the menu will know the difference, but will you?

## 1.13 Changing the item action.

With the "action" of an item is meant: the things the HCMenu will do when a user selects the item. There are two types of action.

When the item is a "SubMenu" item, the action will be the loading and displaying of an other menu. The action will contain the name and path to the menu that should be loaded. Like: "S:MenuData/Sub.menu".
For a itemm of type "Normal", the action will be a list of commands that should be executed when the item is selected by a user. This list is shown in the listbox in the window.
Since a SubMenu item only loads one other menu, it will never have a list of actions!

A new command is added to the list by clicking the "action" stringgadget

and entering the command in the stringgadget. When the typing is done
pressing <enter> will add the command to the list. Use the arrow buttons
to position the command within the list.
Editing a command is even more easy. Click the command that should be
edited. The command will now be visible in the stringgadget below
the list. Edit the command and press <enter> when ready.

The action button
-----------------
The button at the left of the stringgadget can save a lot of typimg.
Since the "Action" depends on the type of item, the function of the
"action" button also depends on the item type. Therefore it is
recommended to select the
itemtype
 before using the actionbutton.
When the action button is clicked a filerequester appears.

The menutype is "SubMenu":

The filerequester will use some settings from the
settings
 window. The
directory shown is the one selected with
FilePath
 and the files shown
were filtered using the
Pattern
 set in the settings window. When these
settings are right the requester will show menufiles. By selecting one
the name of the menu (and its path) will be copied into the stringgadget.

The menutype is "Normal":

Now the requester will show a directory and all its files. Now the file
can be selected which should be executed when the user clicks the item.
The checkboxes below the stringgadget are used with "Normal" items only.
With these boxes the exact action made can be changed:
- When the "Cd Path" box is checked an additional "cd" command is added
  before the "run" command. When this box in not checked the complete
  filename (with path) is used for the run command.
- With the "FileName" checkbox is selected that a file should be
  selected. When the "run" checkbox is not checked this one will be
  disabled!
- With the "Run" checkbox is selected that a "run" command should
  precede the filename. Since a filename is expected checking this
  box will disable the filename box.

Example:
When only "Cd Path" is checked only the directory of the requester is
used to create a CD command without running any file. Checking only
the "FileName" box will give a filename with path. Selecting both the
"Cd Path" and "Run" boxes will give:
  "cd <directory>"
  "Run >NIL: <filename>"
Selcting nothing (= expecting nothing) and clicking the action button
is totally useless!

PS:

To warrant compatibility with future versions of the HCMenu it
is recommended to start the ItemAction with a number or letter and
not with a special character. Characters like the at ("@") and the
percent sign are used in special functions and I don't know which
other characters will be used in future.

## 1.14 Adapting the Itemtype.

The itemtype determines greately the function of an item, and thereby the
look and function of the total menu. This is possible since different
types of items may have different colors.

These are the types available in this version of the editor:

"Normal"
--------
This is the oldest itemtype existing. This type is for the items which
start applications and executes commands.

"SubMenu"
---------
An item of this type will load a new menu. There are no commands to be
executed. The name of the menu to be loaded must be specified in the
action stringgadget.

"Invisible"
-----------
An invisible item is not exact an item, but more the missing of item.
By specifying an Invisible item you specify that you don't want an item
on that certain position. When an existing item is changed to the invisible
type, it looses all other information since an invisible item doesn't
need any information.
When multiple gadgets should be made invisible you maybe want to use the
function "Make Invisible" from the
EditMenu
in the
MenuBalk
.

With the above mentioned types of items it should be possible to make a
neat and handy menu. It is very handy to put all submenus together and
to seperate different groups of items with invisible items.

## 1.15 The menu-preferences window

This window is used to adapt the menuwindow to ones taste. All adaptions possible with this window only affect the look of the menuwindow. When the window is closed using the closegadget in the upperleft corner, changes may be lost!

These are the things to adapt....

Item/Line
---
With this slider it is possible to set the number of items on a line. The minimum (and default) is one item on a line. The maximum is 15. On a standard 640 pixels wide screen this looks terrible, but on a high-end monitor using amazingly high resolutions I think it must be possible.

MenuWidth
---
This slider sets the width of the menuwindow, and indirect the width of the items used. The width of the items is also dependant of the number of items on a line.
The minimal window width is 67. This value is defined after a long study with feuroistical and perestonidial effects taken into account (In english: I needed a minimal, tested 67 and liked it....). The maximal value is equeal to the width of your workbench screen including "overscan".

Extra Size
---
With this "slider" the heigth of the item can be increased. This can be done with a value between nothing and 20 (additional) screen lines.

MenuColor
---
With the nice colored cubes it is possible to select the color used for the names in the submenu items. When these items are not visible at all it is highly possible that the textcolor is the same as the background color!

ItemColor
---
With these cubes the user is capable to set the color for the other item types (read "normal" types for this version of the editor...). By selecting a different color compared to the color chosen for the submenu-items a better separation may gbe achieved.

Font
---
This Gadget makes an Asl-font requester appear. In this requester the user may select the fount which will be used in all items. The name and size of the active font are shown in the prefswindow.

```
Borders
-------
With this checkbox the user may select if the items should have 3D
borders or not. Invisible items don't have borders, but in the editor
they are selectable anyway.



Now all other stuff in the window:


Help
----
To open this text.


Use+Close
---------
The menuwindow will re-shape and the prefswindow will be closed. Any
problems with sizes etcetera will be notified.


Use
---
The menuwindow will re-shape. Eventual problems with the sizes etc. are
made visible in the prefswindow directly. Offcourse the prefswindow
stays open.


Save+Close
----------
The new snapshot for the HCMenu will be writen to disk. The menuwindow
will also be changed and the prefswindow will be closed.
```

## 1.16 Functions available in the MenuBar.

```
                The functions available in the menubar are devided in different ↩
                   groups.
These groups try to match Commodores styleguide as close as possible.

On the menubar these topics are found:


                 Project
                 - Information, Files, Settings, Quitting.


                 Edit
                 - Functions to edit the MenuWindow.


                 Automatic
                 - Functions whith an automatic feel.
```

Windows
- Opening and finding of windows.

PS: When the word "menu" is used in this help, the HCMenu-file which
is subject to editing is meant. This in oposite to the menubar
in the screenbar.

## 1.17 The Project menu on thr MenuBar.

The project menu contains the basic functions of the application. These
functions are: the loading and saving of files, adapting the editor to
ones taste, getting (system)information and leaving of the application.

These are the topics in the project menu:

New
---
Start with a new menu. The menu which is loaded at the moment will
be discarded and an empty menu will be made. This is the same for
a writer who throws away the page and starts over with a new blank
page.

Open
----
Load an existing menu. The menu which should be loaded is selected
using a filerequester. The menu already present will be discarded when
the existing menu is opened. This function acts the same as the Gadget

Open
from the
ToolWindow
.

Write
-----
This function will write a menu using the name it had when it was
opened. An other name for this function could be "Update". When the
menu is new, the editor will invoke the "Write As" function which
is described below.
This function equals the
Write
function from the toolwindow.

Write As
--------
This function uses a filerequester to write the file. This makes it
possible to give a new menu a name, or to write the menu using an
other name than the name it was opened with.

```
Import menu
-----------
With this function it is possible to load menufiles from other menu
applications. These files can be edited and saves as normal HCMenu
files. Se also the information about
                import
                .


Settings
--------
This calls the
                SettingsWindow
                . With this window the HCMenuEditor may be
adapted to the users taste.


Help Me
-------
Starts this online-help about the MenuBar.


Information
-----------
Informs about the system and the editor. The system information only
gives the status of your memory.
The editor information gives the name of the menu which is opened at
the moment, the number of items in the menu and the amount of memory
consumed by the menu. Also the filelength on disk is estimated.


About
-----
Information about versions, the coder (me) and other (non)sence.


Quit
----
Leaving the Haute-Cuisine Menu Editor. Clicking the closegadget on the
menuwindow does the similar job...
```

## 1.18 Importing other menufile formats.

```
When importing other menu formats I found the item information to be
the most important. Part of this information is the way the items are
positioned in the menu. The size of the items and how they looked
were considered less important. The goal of the importing was to
make other menuformats accesible for the HCMenu, and not to make the
editor useful for other menu systems.

Now is following a list with menuformats known by the HCMenuEditor.
Also the version number of the applications I used and the restrictions
for succesful importing are noted:
```

MenuMaster V1.0

menuMaster is an application made in 1990 by Dave Matthewson. The Menu-
Master does not support submenus and always consists of 32 items. These
items are porsitioned in two rows of 16 items. An itemname is limited
to 35 characters. Every item can execute only one command of maximal
128 characters. An additional feature is that every item can have an
extra text. This text is not used by the HCMenuEditor.
The MenuMaster file is always stored in the root of the bootdisk and
is named "MenuMaster.data".

Menuts V1.71b

Menuts is an application made in 1993 by ing. M. van Aalten. Development
of both HCMenu and Menuts were almost parallel. This was caused by me
and ing. van Aalten informing each other of new functions incorporated
in our menusystems. This made both applications a bit similar.
Although, since some things work very different in a menutsfile, both
menu types are not 100% convertable into the other type.
A menutsfile may be placed anywhere on the disk and ends with ".List".
The main menu is called "menuts.list".
Menuts supports submenus and multiple commands per item. These things
are converted completely to their HCMenu equals. The size of an item
button is always 128 pixels. The window size is always a multiple of
128 pixels. Itemnames are maximal 12 characters large. Since the
position and the size of a menutsmenu is defined completely different
compared to the HCMenu differences may occure when importing.

## 1.19 The Edit menu on the MenuBar.

Normally the edit menu contains the functions where the ↩
application
is all about, although, in this version it is true.
The editmenu contains functions with which the user is able to edit
a HCMenu, which was probably the reason he started the editor in the
first place...

The functions always act on
selected
items!

The functions in "Edit":

Edit Item
---
This function calls the
ItemEditor
. The
Edit
Gadget in the
ToolWindow
has the same function.

Make Invisible
--------------
This function turns all selected items into "invisible" items. Items which are invisible are not really items. They can better be seen as seperation spaces achieved by removing the items from that spot.
With the help of this function the user doen't have to use the ItemEditor to make the items invisible manually. Remember: Invisible gadgets don't contain information, therfore all information will be discarded when a item is turned into an invisible item.

Delete Item
-----------
Remove all selected items from the menu. And gone is gone, a deleted item will never return! {:-(
This function is the same as the
Delete
Gadget in the toolwindow.

Insert Item (Up)
----------------
Insert an empty item BEFORE the selected item(s).

Insert Item (Down)
------------------
Insert an empty item AFTER the selected item(s).

Cut Item
--------
Move an Item to the clipboard. With move I really mean Move: The item is deleted from the menu and stored on the clipboard. With "paste" (see below) the item is placed on the menu back again.
See also
Cut
from the ToolWindow.

Copy Item
---------
Copy an item to the clipboard. The item now exists in both the menu and the clipboard. With "paste" the item can be placed on the manu.
See also
Copy
from the ToolWindow.

Paste item
----------
This function also knows an Up and Down version. This functions the same compared to the insert Up/Down functions.
With paste the item(s) at the top of the clipboard are moved to the menu.
See also

Paste
op het ToolWindow for aditional
information.

## 1.20 The Automatic menu on the MenuBar.

This menu contains functions which actions could also be performed using the functions in the "Edit" menu. But using these functions will save in time:

Build Menu
---
This item calls the
BuildWindow
which is capable of generating a menu from a directory or from a entire disk!

Sort Menu
---
Sorteren the menu as defined in the
SettingsWindow
.

CleanUp Menu
---
Remove all unused items (items which are visible, but don't have an "action"). Gone is gone!

Use Capitals
---
This function will convert the first letter of the itemname into a capital. This makes menus very nice...

## 1.21 The Windows menu from the MenuBar.

With this menu the you are able to open windows. When a particular window is already open, use this menu to get the window on top of thye screen. This is handy when the toolswindow is somewhere behind an other window.

These are the windows ready to be controlled:

MenuWindow
---
This window should *always* be open. The only thing this item can do is to get it on top of the screen again.

```
PrefsWindow
-----------
This opens the
                PeferencesWindow
                 which controls the look of the HCMenu.


ToolWindow
----------
This opens the
                ToolWindow
                 which contains some very handy functions.


ClipBoard
---------
This will open the
                ClipBoard
                . With the help of this windo every cut, copy
or paste option is controlled more easily.
```

## 1.22 The settings Window.

```
                The settings window is meant to set all kinds of things concerning ↩
                    the
MenuEditor itself. Adjustments made in the settings window only affect
the editor and not the menu which is edited.
The settings are written to a so-called
                settings-file
                . This file is only
used by the editor and not by the HCMenu. Storing the current settings is
done by clicking the "Save" gadget at the left bottom of the window.
By clicking ht "Use" gadget, the current settings will be used in the
editor, but they will NOT be written to file.
The "Cancel" gadget will discard all changes made in the settings window
and set the editor back in the previous state.

The elements of the settings window:


                 File Path
                  -The search path to the Menu-File's.

                 Pattern
                  - The pattern of the Menu-File's.

                 Log File
                  - The name of the log file.


                 Update Snap
                  - Should the snapshot be updated?
```

```
                Sort: Submenu
                 - How to sort the menu?

                Messages to:
                 - Where to show the messages?

                Requesters:
                 - Suppressing of requesters.

                Crunching:
                 - How to compress a menu-file?

                CrunchBuffer:
                 - How large is the compress buffer?

These elements are not found in the list above:

  Custom Screen, Prefs Window, Tool Window and ClipBoard.

These elements indicate how the Editor will startup the next time. The
element "Custom Screen" indicates if the Editor should run on its own
(custom) screen, or on the WorkBench.
The elements "Prefs Window", "Tool Window" and "ClipBoard" indicate
if the equally named windows should be openened automagical when
starting the Editor. The status of these two elements depends also on
the status of the windows when opening the settings window.
```

## 1.23 The settings-file and its contents.

```
                The settings file is found in the "ENVARC" directory and is called
"HcMe.prefs". The file is protected by a checksum so when problems
with the file occur, the user will be alerted. In that case the Editor
asks the user whether to use the defaults (recommended) or to use
the loaded values anyway (which may crash the Editor).
When there is a new version of the HCMenuEditor installed, it is
possible that it will give a checksum error. In that case there is
no problem by selecting the loaded settings instead of the defaults.



Indirect settings
-----------------

These are the settings where the user is not aware of. These settings
include sizes/positions of windows (including ASL-windows). These
settings make sure that the GUI will look the same every time and
the user may decide where to put this window and where to put that.



Direct settings
---------------

These settings are made by the user and he is aware of it. These are
settings made in the settings window and the
```

```
                build window
                . Things
like file paths and file patterns for instance.
```

## 1.24 Setting the search path in the settingswindow.

```
The "File Path" indicates where the Editor should store/find the
menu-files. This way the Editor dows not hace to ask for a
directory when loading/writing menu files.

There are several ways to set the File Path. The Editor remembers
where a loaded file came from. When saving the settings this path
will be saved as the File Path. This is an indirect method.

In the settingswindow the user can alter the File Path in two
different ways:

- Using the String Gadget. It is possible to type the path in
  thw string gadget. Whithout mistakes offcourse.
- Using a Als-requester. At the left of the string gadget there is
  a "normal" gadget. By clicking this gadget an Asl-requester
  pops up. This requester will show dirictories only. No files!
  Use it to select the desired directory and select "Okay".
```

## 1.25 Setting the search pattern.

```
With the help of the search pattern the user is able to filter the
files shown in the filerequester. The usage of the pattern is equal
to the usage of the pattern with the "Dir" command.

For the pattern WildCards can be used. Below some examples of
patterns are shown: (see also the AmigaDOS manual)

? Use a random character. eg:"1234?67". Now the files called
  "1234567" and "1234A67" are valid.

# "One or multiple". Using the pattern "#E" will show all files
  with only the "E" as mame (these include "E", "EEE" and "EEE").
  In combination with the questionmark the sign "#?" appears.
  This means "one or multiple random characters" and will show
  all files in the directory. The pattern "#?.menu" will show
  all files ending with ".menu". The "#?" pattern is equal to
  the asterisk ("*") on older systems.

~() "Not". The files with the pattern between the braces must NOT
  be shown (supressed) instead of just shown. The pattern
  "~(#?.info)" shows all files except the ones ending on ".info".

( | ) "Or". Two patterns separated by a vertical bar means that it
  means for both patterns: ("#?.menu"|"#?.bak") shows both the
  files ending with ".menu" and the files ending with ".bak".
  The pattern "~(#?.info|#?.bak)" suppresses files ending with
```

".info" and files ending with ".bak".

## 1.26 Setting the name of the logfile.

When using the HCMenuEditor it will produce messages. When in the settings window the option
Messages to
is set to file messages, the
messages will be stored in the file set here (eg "RAMme.Log").

The Editor produces different kinds of messages. The most important
are the messages when something went different than expected. Also when
sumething *did* succeed, this is also messaged ("File written successfully").

Some of the messages can get quite big. An example for this are the
messages of the
BuildWindow
when searching a directory recusively.
Since these messages come by very quick it can be handy to view
the later in the log file.

## 1.27 Updating the local snapshot.

Some things can be snapshot per individual menu instead of one setting
for all menus. These "things" are:

- Window positions (zoomed and normal).
- Window width.
- Number of items per line.
- SubMenu/Item color.

These options are set using the
prefwindow
. By switching on "Update
snap" in the settings window (default) the Editor will write a local
snapshot with every menufile written.
Writing the general snapshot is only possible by using the prefswindow.

## 1.28 Set the sorting algorithm.

Menus are sorted by the function "Sort Menu" in the "automatic" menu
on the menubar. This is very handy after the automatic menu generation
using the
BuildWindow
.

These options are available:

```
First
-----
All submenu's are in the top of the menu, and are followed by the other
items. The way the submenus linedup is not affected, this is also the
fact for the normal items.


First+Alpha
-----------
All submenu's are in the top of the menu and sorted alphabetically on
the name. The normal items follow also sorted on name by alphabet. If
a name starts with "A" or "a" doesn't matter.


Last
----
All normal items are sorted to the top of the menu. The other items
(the submenus) follow. The way the normal items linedup is not affected,
this is also the fact for the submenus.


Last+Alpha
----------
The oposite of "First+Alpha": all items sorted alphabetically on name
on the top. The other items follow also sorted alphabetically on name.


Mixed
-----
All items are sorted alphabetically on name. Submenus and items are
mixed through each other.
```

## 1.29 Setting the message medium.

```
                As most software the HCMenuEditor has things to mention the user.  ↩
                   The
medium used for these messages can be selected here. These are the
options:

Screem  - All messages go to the screen.
LogFile - The messages are written to the file which is set in the

                LogFile
                 option of the settings window.
Scr&Log - The messages go to both the screen and the logfile.
None    - The Editor shits its big mouth. No messages at all.

For high-priority messages like errors the HCMenuEditor uses a

                Requester
                 also. Therefor it is no problem to
select the option "None". Problems are shown via requesters.
```

## 1.30 The suppressing of requesters.

When something unespected happens (disk errors) or when the user selects a "dangerous" funcion (lik delete) the editor reacts with a requester.
Since not all requesters are of equal importance the user can select from which level the requesters may be shown, and what requesters may be supressed:

All - The safest choise. This option is recommended for less experienced users. A lot of options will ask the "Are you sure?" question before a operation takes place.
Some _ For the more experienced user. The "Are you sure" questions don't come up anymore. Only errors will be shown.
None - Only for the Die-hards. NO requester whatsoever will come up. Even if you format the harddisk nothing will be said...

PS: By selecting None for the requesters and "None" for the messages the HCMenuEditor is unable to give ANY indication of errors! Use it at your own risk!

## 1.31 Setting the crunch efficiency.

This option was implemented to make things complete. Compressing menu-files does not make them kilobytes smaller. At the other hand: since the files are small the time needed for crunch/decrunch is almost not noticable.

The crunch efficiency is a compromise between the time needed for the crunching process and the file size at the end. The more efficient the algorithm, the smaller the file will get. But that will take more time compared with a algorithm with less efficiency. But that algoritm will produce larger files.
Since menufiles are small, setting the efficiency is almost meaningless. The difference in size is small (if there is any). And the time needed seems to be the same also.

One of the options is "None". Crunching with no efficiency is the same as turning crunching off. And that is what the option does. The benefit of non-crunched files is that the files are written as text-files. These text files can be edited by using "Ed:".
When "None" is selected setting the buffersize is disabled. This option has no meaning when the files will not be crunched.

## 1.32 Setting the size of the crunch buffer.

As with the
efficiency
setting, setting the size of the crunch buffer
affects the time needed to crunch the files.
The larger the buffer the smaller the amount of time needed. Since the
smallest crunch buffer is larger than the average menufile, this option
will not make much difference...

## 1.33 The ToolWindow

The tool window is a small window that is meant to make functions ↩
that
are used much more accessible. The toolwindow can be opened via the

Windows
menu (option "Tool menu")on the menubar of the screen.
The toolwindow has the following functions:

Open
- Open an existing menu.

Write
- Write the active menu.

Edit
- Change items using the itemeditor.

Delete
- Remove selected item(s).

Insert
- Insert a new item.

Cut
- Move an item to the clipboard.

Copy
- Copy an item to the clipboard.

Paste
- Insert an Item from the clipboard.

<-
- Insert / Paste direction.

- Number of Inserts / Pastes.

Using the
SettingsWindow
it is possible to define if the toolwindow
should be opened when starting the editor or not.

## 1.34 The ToolWindow, Gadget: Open.

With the "Open" Gadget an existing menu can be opened. After selecting
the open option an Asl file-requester will pop up. With the requester
the menu to load can be selected.

When the "Cancel" Gadget is selcted the requester will disappear and
nothing will be changed to the editor, or the menu that is currently
(still) loaded.

When a file is selected and the "Okay" Gadget is clicked the editor
will load the file selected. When this file is indeed a correct menu
file the menu will be displayed.

## 1.35 The ToolWindow, Gadget: Write.

With the Gadget "Write" the you can save your work...

When the option write is activated a last requester will appear,
asking to save or not ("Cancel").
IF the "Write" Gadget is selected the active menu will be saved
using the name with which it was loaded using the "open" option.

When the menu is new and does not have a name yet a file-requester
will pop up asking for the save-name and position.

## 1.36 The ToolWindow, Gadget: Edit.

With the "Edit" Gadget menu items can be adapted to your needs...

The Edit Gadget will open the
ItemEditor
. With this editor it is possible
to give an item the name and function you want. The edit Gadget will only
work for items that were
selected
.

The ItemEditor will be opened once for every item selected. When multiple
items were selected the editor will open "multiple" times. It is not
possible to select a group of items and give them the same function.

For this task it is better to use
Copy
and

Paste
.

## 1.37 The ToolWindow, Gadget: Delete.

With the "Delete" Gadget items are removed from the menu.

Items that are removed from the menu are gone. There is no way of getting them back as there is no "undelete" nor "undo" option within the HCMenu-Editor.

The delete Gadget will only work for items that were
selected
.

## 1.38 The ToolWindow, Gadget: Insert.

With the "Insert" Gadget empty items are added to the menu.

The Insert function will add empty items above or below any item that is

selected
. If items are added above or below is selected through the

direction gadget
.
This direction gadget is found at the bottom of the toolwindow. It will show an arrow pointing up, or an arrow that points down. When the arrow points up, the empty items are added above the slected item.

The empty item is always of type "normal" and does not have a name or function/commands. This can be changed with the
ItemEditor

## 1.39 The ToolWindow, Gadget: Cut.

The "Cut" Gadget moves items from the menu to the clipboard.

"Move" means indeed: remove from the menu and add it to the clipboard. With the
Paste
Gadget the items on the clipboard can be placed back in the menu,

The Cut Gadget will only work for items that were
selected
.

When multiple items that were sleced they will be transferred as a group. When items are "Paste"d back to the menu a group will be stored as a

whole in one Paste action.

## 1.40 The ToolWindow, gadget: Copy.

The "Copy" Gadget copies items in the menu to the clipboard.

The Copy Gadget will only work for items that were selected.

When the copy function is finished the menu will show no differences. This in oposite to the clipboard. All items selected will now be available on the clipboard also.

When multiple items were selected, these items are copied as a group. All items that were copied in one go are member of the same group. When a Paste is invoked all items of one group will be added in one stroke.

## 1.41 The ToolWindow, gadget: Paste.

The "Paste" Gadget moves items from the clipboard to the menu.

The Paste function will move the items above or below any item that is selected. If items are added above or below is selected through the direction gadget.

This direction gadget is found at the bottom of the toolwindow. It will show an arrow pointing up, or an arrow that points down. When the arrow points up, the items are added above the slected item.

Paste will always start with the last item added to the clipboard. The Paste function will always move a group of items to the menu. A group is a set of items that was added together to the clipboard.

## 1.42 Het ToolWindow, the direction gadget.

With the direction gadget the user shows if an item should be added above the sleceted item, or below. The item added can be a new one (from the Insert function) or one from the clipboard (with the Paste function).

```
The gadget has two options: /\ <- - Above (or at the left).
        \/ -> - Below (or at the right).

Above or left
-------------
All items added will be added before the selected item. When the menu has
only one column this will be above the selected item. When there are
multiple items on a row it will be at the left of the slected one.


Below or right
--------------
All items added will be added after the selected item. When the menu has
only one column this will be below the selected item. When there are
multiple items on a row it will be at the right of the slected one.
```

## 1.43 The ToolWindow, Number of Pastes/Inserts.

```
With the slider at the bottom of the ToolWindow the number of items
to paste/insert is selected.

Example: When the slider is set to 4 and tht user selects "Paste",
four items are pasted from the clipboard at every selected item. When
there are not enough items in the clipboard a warning is displayed.
```

## 1.44 The automatic menubuilding Window

```
                This window is ment to create a menu from a given directory. Since ↩
                    not
all files will be interesting for including in a menu, the user is able
to make a selection on forhand.

All selections and settings in this window are stored in the settings
file. With the
                SettingsWindow
                 this file may be written.

By checkmarking the checkboxes at the left of the stringgadgets the
stringgadgets are enabled. When a filetype is checkmarked, it will be
used during the menu creation.
These types are available:
                  Execute

                  SubMenu

                    IFF

                  .......
                The Pattern-stringgadgets.
-------------------------

At the right of the checkboxes the stringgadgets are found. The first
```

```
is used for defining the searchpattern. How to use these patterns is
described in the section
                Pattern
                 from the settingswindow.
The pattern is ued to define which files should be taken into account.
(All files ending .IFF for the IFF filetype for instance)


The Prefix-stringgadgets.
-------------------------

The prefix enables the user to add commands to the ACTION of a menuitem.
The prefix will be used for all files found from this particular type.
For a IFF file the prefix "SYS:Utilities/MultiView" could be used. This
will produce a menuitem which starts MultView to show the IFF-picture.


Selecting the searchpath.
-------------------------

In the path-stringgadget it is possible to enter the directory from
which the build has to start. It is also possible to click the Path
Gadget which will activate a directory-requester from which the user
may select the desired path.


Other checkboxes.
-----------------

Beneath the Path gadgets another two checkboxec nay be found. The first
one is the "New Menu" checkbox. When it is checkmarked a new menu will
be build instead of added to the existing menu.
The second checkbox is the "Recursive" box. This checkbox determines if
subdirectories found in the searchpath should also be entered. When the
box is not checkmarked only the path in the Path-stringgadget will be
scanned.


The Buffer cycle-gadget.
------------------------

With this gadget the size of the directory buffer used is determined.
When the recursive algorithm is used and the directory structure is
large, great amount of memory is used. Using a small buffer will slower
down the scanning, but will reduce the memory consumption.
Since a small buffer doesn't affect the scanspeed significantly, this
gadget is the first to go when I need it's window-space...
```

## 1.45 The Execute filetype.

```
The execute filetype consists off all executable files/commands. The
editor checks if the file really is executable. To speed up the scan-
ningprocess the checking is simple and not perfect.

Some systemfiles are executable without any need. Examples are .Font
```

and .Library files. The fonteditor will find these files executable.
With the help of the pattern the user may exclude these types of files
to prevent them of being used in the menu: " ~(.font|.library) ".

The action
----------

The action of an executable is build as follows:

  cd <Path>; <Prefix>; Run >NIL: <File>

The prefix is specified by the user and may consist of multiple
commands separated by a semicolon ";". The <Path> is the path
where the file was found during the scan.

## 1.46 The SubMenu filetype.

By scanning a disk for menus, a main menu may be created. Since a menu
is just a tekstfile no checking is done.

The pattern used for scanning menufiles is the same as the pattern
specified in the
Settings
.

Since a item of the type "SubMenu" only needs the name and path of the
menu, no prefix is needed.

The action
----------

The action of a submenu is build as follows:

  <Path>/<MenuFile>

Offcourse, the item created is of the "SubMenu" type.
<Path> is the place where the menu was found during the scan.

## 1.47 The IFF filetype.

The editor is also able to scan for IFF files. The editor only checks
if the files are of the type IFF. It does not check if a possible file
is a picture, sound, text or ....
Since the HCMenu doesn't use the IFF-files itself, a IFF viewing tool
should be specified. This is done using the prefix.

The action
----------

The action of a IFF-file is build as follows:

```
  <Prefix> <Path>/<IFFFile>
```

When the user specified the prefix as: "SYS:Utilities/MultiView" the resulting action would be:

```
  Sys:Utilities/MultiView DH1:Grafiek/Plaatjes/JSW_Background2
```

## 1.48 The ..... filetype.

That's what I call a strange filetype. This filetype may be specified by the user. No checking is performed offcourse. With the help of the search-patterns the user is able to pick the right files.

### The action

The action of a ..... is build as follows:

```
  <Prefix> <Path>/<.....File>
```

When the user specified the prefix as: "SYS:C/Delete" and the pattern as #?.info", the resulting action would look like:

```
  Sys:C/Delete DH1:Grafiek/Plaatjes/JSW_Background2.info
```