

**Emperor**

Copyright © Copyright2000-2001 by Matthias Gietzelt

<b>COLLABORATORS</b>
----------------------

	<i>TITLE :</i> Emperor		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY		July 31, 2024	

<b>REVISION HISTORY</b>
-------------------------

NUMBER	DATE	DESCRIPTION	NAME

# Contents

<b>1</b>	<b>Emperor</b>	<b>1</b>
1.1	Documentation of Emperor . . . . .	1
1.2	Introduction . . . . .	2
1.3	Copyright . . . . .	3
1.4	Systemrequirements . . . . .	4
1.5	Installation . . . . .	4
1.6	Getting started . . . . .	5
1.7	Operating . . . . .	5
1.8	Menu . . . . .	6
1.9	Mainwindow . . . . .	7
1.10	Mainwindow - Reaction - page . . . . .	8
1.11	Mainwindow - Requests - Page . . . . .	12
1.12	Mainwindow - Menu - Page . . . . .	13
1.13	Mainwindow - ASL - Page . . . . .	13
1.14	Mainwindow - Locale - Page . . . . .	13
1.15	Mainwindow - Array - Page . . . . .	13
1.16	Mainwindow - GlobVars - Page . . . . .	13
1.17	Mainwindow - Windows - Page . . . . .	14
1.18	Mainwindow - GadTools - Page . . . . .	14
1.19	Mainwindow - Interconnection Map - Page . . . . .	15
1.20	Programminghelp . . . . .	15
1.21	Input message handle . . . . .	16
1.22	predefined variables . . . . .	16
1.23	Macrofunctions . . . . .	17
1.24	Function Emperor_CloseWindow . . . . .	17
1.25	Function Emperor_OpenWindow . . . . .	18
1.26	Function Emperor_IconifyWindow . . . . .	20
1.27	Function Emperor_UniconifyWindow . . . . .	21
1.28	Function Emperor_ChangeWindowPosition . . . . .	22
1.29	Function Emperor_ChangeWindowSize . . . . .	22

---

1.30	Function Emperor_SetWindowBusyPointer . . . . .	23
1.31	Function Emperor_SetWindowTitle . . . . .	24
1.32	Function Emperor_SetScreenTitle . . . . .	25
1.33	Function Emperor_GetMenuItemAttr . . . . .	25
1.34	Function Emperor_SetMenuItemAttr . . . . .	27
1.35	Function Emperor_ActivateGadget . . . . .	28
1.36	Function Emperor_QuitFunc . . . . .	28
1.37	Function Emperor_RethinkLayout . . . . .	29
1.38	Function Emperor_RefreshGadget . . . . .	30
1.39	Function Emperor_GetGadgetAttr . . . . .	31
1.40	Function Emperor_GetGadgetAttrComplex . . . . .	32
1.41	Function Emperor_GetGadgetDisabledAttr . . . . .	33
1.42	Function Emperor_GetGadgetReadOnlyAttr . . . . .	34
1.43	Function Emperor_SetGadgetAttr . . . . .	35
1.44	Function Emperor_SetGadgetAttrComplex . . . . .	36
1.45	Function Emperor_SetGadgetDisabledAttr . . . . .	37
1.46	Function Emperor_SetGadgetReadOnlyAttr . . . . .	38
1.47	Function stringtoint . . . . .	39
1.48	Function inttostring . . . . .	40
1.49	Function stringlength . . . . .	40
1.50	History . . . . .	41
1.51	FAQ . . . . .	43
1.52	known bugs & future plans . . . . .	44
1.53	Author . . . . .	44
1.54	Expression of thanks . . . . .	45

# Emperor

<u>/*****/</u>	<u>/** </u>	<u>/** </u>	<u>/***\</u>	<u>/*****/</u>	<u>/***\</u>	<u>/***\</u>	<u>/***\</u>	<u>/***\</u>
/**____/	/*** /***		/*__**\	/**____/	/*__**\	/*__**\	/*__**\	/*__↵
**\								
/**/____	*/ ** *		*/__/*	/**/____	*/__/*	*/__/*	*/__\*	*//____↵
/*								
/*****/	*/ _/*		/*****/	/*****/	/*****/	/*	/*	↵
/******/								
/**____/	*/	/*____/	/**____/	/*__**____/	/*	/*	/*__**____/	
/**/____	*/	*//	/**/____	/**/____	*/\*\	\*\____/*/	*/\*\	
/*****/	*/	*//	/*****/	*//	\*\	\*****/*/	*/\*\	
/____/	/	/	/____/	/	\*\	\____/ /	\	

Available at [www.aminet.de](http://www.aminet.de) ! (Drawer: dev/c)

How does Emperor make working more easily ?

History	What's the story ?
FAQ	Which questions were asked frequently ?
known bugs	Which bugs does Emperor have ?
expression of thanks	Whom I want to thank very much ?
Author	Who made it ?

## 1.2 Introduction

Please support the further development of this software and mail me your opinion, problems und criticisms, because the programming costs a lot of time (but it was a pleasure!).

Thank you for decrunching and the will to testing the program-packet. This program is for "revolutioning" the software-development on Amiga.

Object-oriented programminglanguage (OOP) is maybe a bit too high for that program, because there are "real" C/C++-sourcecodes generated and an external C/C++-Compiler must translate it into machine-code. In the Amiga- sphere

there are only less applications like CanDo, Storm-Wizard and so on, but one day I have seen the program "Delphi" on the PC :( ! This OOP is exclusive controlable in Pascal (that set's me up) and it has millions of configuration-possibilities. I didn't knew such things on Amiga, and so I made it by myself.

I want to make parameting of the Graphical User Interface (GUI) and its macro- elements as easy as possible. Following standard-GUI-elements are implemented:

- \* Windows
- \* OS2.0 Gadgets (GadTools)
- \* OS3.5 Gadgets (Reaction)
- \* Requests (EasyRequest)
- \* Menus (NewMenu)
- \* ASL-Requests (File, Font, Screenmode)
- \* Locale (Translations)
- \* Function-macros, which makes programming under OS3.5 more easily

With Emperor you are generating a complete C/C++-sourcecode, which only must be compiled by an external compiler (e.g. StormC from the developer CD2.1). There are many different files, which are including following data:

PROJEKTNAME.c	Programsourcecode (compileable)
PROJEKTNAME.cd	Catalog-rawdata
PROJEKTNAME.h	own sourcecode
PROJEKTNAME.project	projectfile for Emperor
PROJEKTNAME.¶	Storm-projectfile
PROJEKTNAME_SPRACHE.ct	translated catalog in rawdatas

Aim of this program is to assure software developer to create their programs (its sourcecode)

by Emperor, because the making of software (opening of windows, file- and fontrequests, menus etc.) is often equal and must not longer copied by programmers from other sourcecodes. Emperor decrees about one or more macros for every GUI-element. Like the menus: here you must, for creating the complete Project-menu (open, close, save, print etc.) and its translation, (in Locale-".catalog"-file) only select the corresponding entry in the list. This is also possible with requests e.g. by making a QuitRequest. These are called in the own sourcecode as function "REQUESTNAME();".

But the program should also be for amigans, who never programmed before, and bade an easy way in the programming of AmigaOS-GUIs.

### 1.3 Copyright

Emperor Version 4.1 is copyrighted © 2000-2001 by Matthias Gietzelt  
 Emperor Version 4.1 is Giftware  
 All rights reserved.

You can spread Emperor version 4.1, if your intention is non-commercial.  
 The author takes no responsibility for using the program, and possible happend damage. :)

What is Giftware ?

Giftware you can copy and spread freely, as long as there is made no benefit with the program.  
 If you like Emperor and if you are using it you can send me a little present. - I 'll be very happy about everything. In return I'll offering the programpacket itself and you can have the latest running betaversion of Emperor, long before it will be released to Aminet. I'll send it via eMail or on a disk.

But the program-packet must be complete, which means there must be:

- the program "Emperor"
- the startuppicture "Emperor.pic"
- the prefs-file "Emperor.prefs"
- the install-script "Emperor.install"
- this guide "Emperor.guide" (english and german)
- the german catalog-file "Catalogs/deutsch/Emperor.catalog"
- the french catalog-file "Catalogs/français/Emperor.catalog"
- the raw catalog-file "Catalogs/Emperor.cd"
- the project-directory "Projects/" with some projects
- das scripts-directory "Scripts/" for interactive StormC-connection via ARexx

If you are missing one of these files in the archive, please contact me !

## 1.4 Systemrequirements

- an Amiga ;-)
- Memory 4 MByte; recommended 16 MByte or more
- Workbench version 3.5 or higher
- Processor at least 68020; recommended 68060
- Amiga Developer CD 2.1 of the HAAGE & Partner GmbH
- C/C++-Compiler (from the developer CD "ADCD\_2.1:Contributions/Haage\_& Partner/ ↵ StormC/")
- CatComp (from the developer CD "ADCD\_2.1:NDK/NDK\_3.5/Tools/CatComp")
- xen.font in size 8 installed in "Fonts:"-directory of the Bootdevice, :-)

tested with follow configurations:

- A 1200
  - \* 68060/50MHz (BlizzardPPC-Card)
  - \* 603e+/240MHz
  - \* 100 MB Memory
  - \* BlizzardVisionPPC - graphixcard
- A 4000
  - \* 68040/25MHz
  - \* 64 MB Memory
  - \* CyberVision 64/3D - graphixcard
  - \* FastLane Z2
- A 2000
  - \* 68040/30MHz
  - \* 18 MB Memory
  - \* GVP Series-II SCSI Filecard
- A 2000
  - \* 68040/40MHz
  - \* 1 MB memory CHIP
  - \* 58 MB memory FAST
  - \* Delfina Lite soundcard
  - \* CyberVision 64/3D graphics card
  - \* 10 GByte hard drive space
- A 1200
  - \* 68060/50MHz (Blizzard 1260)
  - \* 1230 SCSI-Kit
  - \* int. IDE HD 60MB
  - \* ext. SCSI HD 3.3GB
  - \* ext. SCSI CD-player
  - \* ext. SCSI ZIP-drive (250 MB)
  - \* AmigaOS 3.9 with BoingBag1
  - \* AGA-screen (800x600 pixel)

## 1.5 Installation

Please use enclosed install-skript.

---

If you want to install by hand, then copy the catalog ("Emperor.catalog") in the directory "SYS:Locale/Catalogs/deutsch" or "SYS:Locale/Catalogs/français";

the program-file ("Emperor"),  
 the startuppicture ("Emperor.pic"),  
 the prefs-file ("Emperor.prefs"),  
 this guide-file ("Emperor.guide")  
 the scripts-dir ("Scripts/")  
 and  
 the projects-dir ("Projects/")

to a place you'd like.

Then add the line:

```
"Assign Emperor: PROGRAMMPATH:"
```

in your user-startup-file and reboot your system.  
 Now you can use the program without an restriction.

"PROGRAMPATH:" should be the path, where you've copied "Emperor".

## 1.6 Getting started

The program can be started by CLI (Shell) without following arguments.

But you can also doubleclick on the icon of the program.

There are a few things, you must pay attention on:

- \* AmigaOS 3.5 installed ?
- \* 4 MBytes of systemmemory must be free on startup
- \* assign on <PATH>:Emperor
- \* RexxMast already started ?
- \* stack at >50000 bytes ?
  - under CLI: type 'stack 50000'
  - under WB: in iconinformations choose a stack of >50000 bytes
  - under DOpus: in menu 'Settings-Environment' choose point 'CLI-Launching' ↔  
                   and  
                   change stack !

## 1.7 Operating

How the menu of Emperor is used...

Menu

How the single windows of Emperor are used...

Mainwindow

Reaction-page

---

```

Requests-page
Menu-page
ASL-page
Locale-page
Arrays-page
GlobVars-page
Windows-page
GadTools-page
InterConnection-page

other windows
Programminghelp
Input message handle

```

## 1.8 Menu

At this place should be the menuentries only listed and short descipted:

```

New          - Deletes the current project from the memory (additional loading of " ←
               template.project")
Delete       - like "New", no loading of "template.project"
Open        - Opens a created project ("*.project"-Datei)
Merge       - Adds a project to the current

Save         (Submenu)
Project save - Saves the current project ("*.project"-file)
Storm-project save - Generates a Storm-project ("*.¶"-file)
Catalog-raw-file save - Generates catalog-raw files ("*.cd" and "*_LANGUAGE.ct"- ←
               file)
Own sourcecode save - Saves the own sourcecode ("*.h"-file)
Catalog generate - Generates the catalog file ("*.catalog"-file)
               ATTENTION ! This is only functionable, if "CatComp" is installed.
sourcecode generate - Generates compilable C/C++-sourcecode ("*.c"-file)
Save as      - Saves the current project under new name ("*.project"-file)
Save as template - Saves the currnet project as template-project ("*.project"- ←
               file)
Clipboard save - Saves the clipboard-content
Save all      - Saves following data:
               (*.c"-file)          generated program-code
               (*.h"-file)          Own sourcecode
               (*.catalog"-file)     Locale-catalog
               (*.project"-file)     Emperor-project
               (*.¶"-file)           Storm-project
               ("*_LANGUAGE.ct"-file) Catalog-translationfile


Print        (Submenu)
Objecttable print - prints the current objecttable
Catalog-raw-files print - prints the catalog-raw files ("*.cd" and "*_SPRACHE.ct"- ←
               file)
Own sourcecode print - prints the own sourcecode ("*.h"-file)
Programsourcecode print - prints compilable C/C++-sourcecode ("*.c"-file)
Print all    - prints following data:
               (*.c"-file)          generated program-code
               (*.h"-file)          own sourcecode

```

---

and the objecttable

Manager - changes to page "Windows"  
 Preferences - opens the window for several program preferences  
 Program preferences - opens the window for several program-specific preferences  
 Information - gives Information about program  
 Quit - quits the program

Cut - cuts the current listentry  
 Copy - saves the current object into the Clipboard  
 Paste - pastes content of the Clipboard into a list  
 Undo - take back last operation  
 Redo - do operation again  
 Delete - deletes the current listentry

\*\*\* Then there are some menuitems, which only can reached from the texteditor. ←  
 \*\*\*

Format text - this item has 2 modi:

\* when the 1.character ( ←  
 row 1) is a space- ←  
 character,  
 Emperor is formatting ←  
 the text in the ←  
 normal way.  
 \* when the 1.character is ←  
 NOT a space-character,  
 the text is formatted on ←  
 the left border

Testwindow - opens a window, which is comparable to the compiled ←  
 project  
 Attributwindow - opens a window, in which you can paramete object-depending ←  
 adjustments  
 Show own sourcecode - opens the internal texteditor  
 Programminghelp - opens the Programminghelp-window  
 Input message handle - opens the Input message handling-window

## 1.9 Mainwindow

The mainwindow carrys as title the name of the current project. It has a statusbar ←  
 ,  
 in which a little help is shown, when the pointer is over a gadget.  
 There are also filing-cards, which contents a determined systemcomponent.  
 So there are 9 different pages:

Reaction-page  
 Requests-page  
 Menu-page  
 ASL-page

Locale-page  
 Arrays-page  
 GlobVars-page  
 Windows-page  
 GadTools-page  
 InterConnection-page

In general the window is equal for all filing-cards.

On the right side there is a list with templates for every component,  
 left from this there are some buttons:  
 with "Add" you can add a modified element to the own list;  
 with "Up"/"Down" you can move the current active entry in the own list up/down;  
 with "Delete" deletes the current active entry from the own list;  
 with "Test" can Emperor show, how your work looks (without compiling)  
 On the left side you'll have an empty list, which should be fill by modified ↩  
 objects.

## 1.10 Mainwindow - Reaction - page

On the gadgets-page there are 3 groups of gadgets:

- \* On the right side there is a list with gadget-templates.  
 If you are choosing an entry, the "Attribute-window for gadgets" is ↩  
 renewed, in which  
 the attributes can be set.
- \* In the middle there are some buttons:  
 with "Add" you can add a modified element to the own list;  
 with "Up"/"Down" you can move the current active entry in the own list up/ ↩  
 down;  
 with "Delete" deletes the current active entry from the own list;  
 with "Test" can Emperor show, how your work looks (without compiling)
- \* On the left side there is an empty list, which later could be filled with ↩  
 the own  
 hierarchical liste of gadget-objects.  
 The entries are at a time the names of the gadgets.  
 An added gadget is put into the next-higher "generation", and it's so  
 an diverted gadget of the previous.

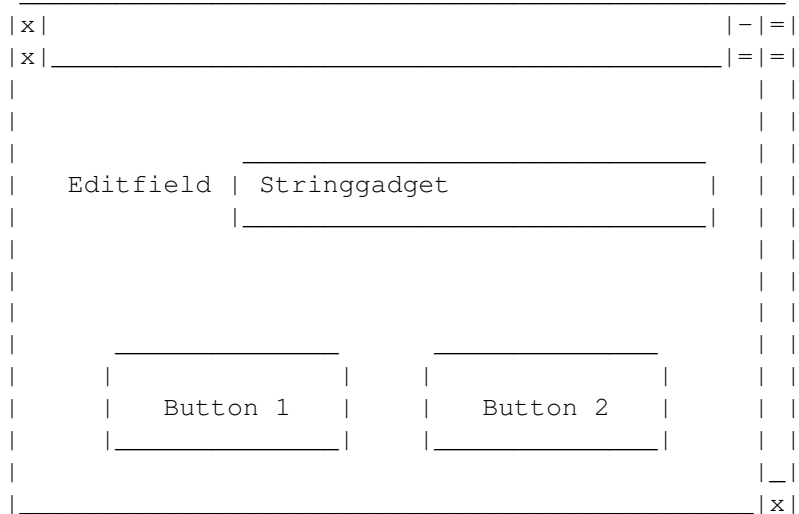
Aim of this "new" kind, to create surfaces is it to took more and more work from ↩  
 the  
 programmer. The positions und sizes are automatically rendered by the system,  
 in relation to the window size.  
 You can influence the arrangement of the gadgets, when you create vertical and ↩  
 horizontal  
 groups (Layouts). Gadgets, which belongs to a layout, must be diverted from the ↩  
 Layout-Gadget,  
 which means that they appear in the list more right and form a new subclass.  
 But horizontal und vertical Layout-Groups can split into new horizontal  
 and vertical Layout-Groups.

Horizontal group means, that all gadgets are horizontal arranged.

Vertical group means, that all gadgets are vertical arranged.

Example 1:

Easy example with one string-gadget (Inputfield) and two buttons.

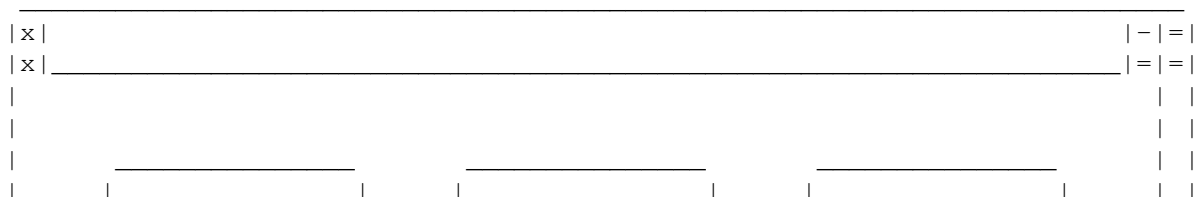


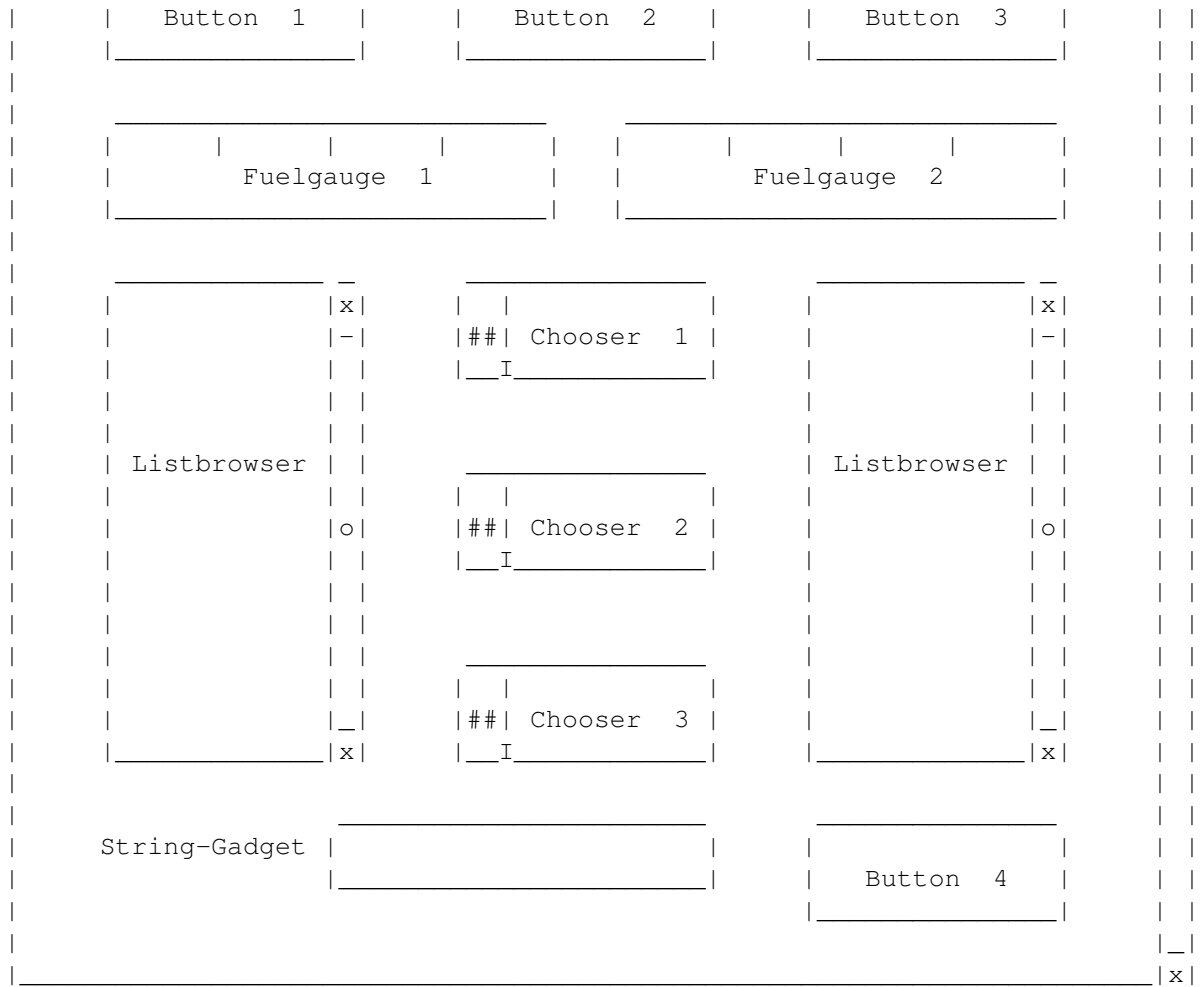
The list for this example must look like that:

```
vertical Layout1    the so-called "Root-Layout" appoints the beginning
|                  of every gadgetlist. In this case it should have
|                  the attribute "Orientation-vertical", because
|                  there are two gadget-groups (1. String-Gadget / 2. two Buttons),
|                  which are layouted vertically.
|
+- String1          the string-gadget is the first visible
|                  object in our list.
|
+- horizontal Layout2  a renewed devide of the space tooks place right here.
|                  First we had devide the whole window in vertical
|                  groups; in one we had the string-gadget and in the other
|                  we are creating a new group, which contents gadgets
|                  that are laying side by side (horizontally  ←
|                  arranged).
|
+- Button1          this is the left Button
+- Button2          and this the right Button
```

It isn't difficult or is it ??  
So, then more extensive:

Example 2:





```

Layout1
|
+- vertical Layout
|
+- horizontal Layout
| |
| +- Button1
| +- Button2
| +- Button3
|
+- horizontal Layout
| |
| +- Fuelgauge1
| +- Fuelgauge2
|
+- horizontal Layout
| |
| +- Listbrowser
| |
| +- vertical Layout
| | |
| | +- Chooser1
| | +- Chooser2
| | +- Chooser3

```



```

| |
+- horizontal Page4
|
+- Listbrowser1
+- Listbrowser2

```

You can generate a "Page" by taking a layout-gadget and turn it into "PageMode" in the attributes-window.

PAY ATTENTION ! Don't forget the first page-gadget below the clicktab !!!!

ad 2 (using images):

Images are special gadgets, which contains e.g. picture informations or system glyphs. ↔

If you want to integrate such an image, you have to pay attention that the image is ↔

derive from the corresponding gadget or you have to assign the image via ↔ connecting

within the attributes-window. But be careful ! You only have got possibilities to derive an image from a button, a layout or a slider gadget !!!

An example may looks like that:

```

horizontal Layout1
|
+- vertical Layout2
| |
| +- Bitmap1
| |
+- vertical Layout3
|
+- Button1
| |
| +- Glyph1
| |
+- Slider1
|
+- Bitmap2

```

SPECIAL NOTES...

## 1.11 Mainwindow - Requests - Page

On the Requests-Page there are entrys in the right list, which generates requests like for information, quit oder errorreports.

An easy click on the corresponding entrys (with the opened "Window-Attributwindow ↔")

you can edit them. Then you can add them into the own list.

There are prepared requests, like the Quitrequest (a dialogwindow, which makes a creating of a request to quit superfluous), whether the program (not Emperor)

really should be quitted, like you know from other applications and tools.

## 1.12 Mainwindow - Menu - Page

In the menueditor in the right list there are 5 entrys; the first three entrys are systemdefinitions (Title, Item, Subitem) -> see AutoDocs;

The other entrys are complete menus. Project stand for the known Project-Menu (open, save, print etc.). Thwe entry must be chosen and added to the own list  
Now you have created the complete menu INCLUSIVE TRANSLATIONS.  
The single entrys can be modified and adjusted to the own taste.

## 1.13 Mainwindow - ASL - Page

Here you can modify the requests for File-, Font- and Screenmode-choosing and test them, until they meet with your ideas.

See AutoDocs !

## 1.14 Mainwindow - Locale - Page

On the Locale-Page you can create catalog for Locale.  
The catalog-Identification will be transported from all parts of the programs (Requests, Menu etc.) into this list.

Only the translation is done right here. For that, you must choose the lauguage ↔ and the catalog-ID, which should be translated.  
Then choose the middle string-field (in the opened attribute-window Menu-"Window- ↔ Attributewindow") and translate the string below the chosen field.

## 1.15 Mainwindow - Array - Page

When you are creating an Array (entrys for Listbrowser, Radiobutton, Chooser and ↔ Clicktabs), you MUST open the attribute-window Menu-"Window-Attributewindow".  
In the Attributewindow there is also a list, which must filled out with entrys of ↔ the Array, before the entry (Name of the Array) is added to the own list in the Mainwindow

## 1.16 Mainwindow - GlobVars - Page

Here you can choose between different variabletypes. Before you add them to your own list, you must make the specific preferences.

The first cyclegadget (from the top) can only be used, when a systemstructure (see AutoDocs) is called and initialized.

The second field specify the variablenames, which are used in the the own sourcecode.

The third is used for the initial value.

## 1.17 Mainwindow - Windows - Page

On this page, you are able to configure your whole windows, which do you wish to implement into your program.

Here you must pay attention, that the current window is marked by "Active". Which means, that the lists of gadgets depends on your current window. I.e., ONLY the content of an "Active" window could be changed anyway.

!!IMPORTANT!!

In the attributwindow you can say set, whether a window should appears in AmigaOS 2.0 or AmigaOS 3.5 - look. If you choose the OS 2.0 - look, you aren't allowed to allocate Reaction-gadgets to it !

!!IMPORTANT!!

## 1.18 Mainwindow - GadTools - Page

Because GadTools only decrees about gadgets, which demands absolute positions, the handling is a bit more complicated (or more logical ? :)).

For creation of an window with GadTools-gadgets (OS 2.0 - look), the window itself must be an OS 2.0 - window. This could be attributed on the Window-page.

Now it is important, that the window, in which the gadget will be positionized, is open.

Now you can choose on the GadTools-page the gadget what you want. And in the second step you have to define position and size by mouseclick (and mousemove) into the window .

Subsequent changings are possible by clicking the marked areas in the edges of a gadget.

Do you want to move or resize a gadget, you can activate it by a simple click.

The left-sided list is deactivated at the GadTools-page.

It is possible to generate a program based on OS 2.0 as well as based on OS 3.5, by supporting of OS 2.0 - windows and gadgets.

## 1.19 Mainwindow - Interconnection Map - Page

Interconnection maps connect gadgets to each other. This connection makes it possible, that different tags and its values between gadgets are exchanged.

I.e. a fuelgauge-gadget is connected with a palette-gadget; there must be clear, which tags are used.

So you can connect tag "FUELGAUGE\_Level" with "PALETTE\_Colour". Everytime the palette is clicked, it will be send its new value direct to fuelgauge.

There are unidirectional and bidirectional maps; unidirectional maps only send informations from primary to secondary gadget, bidirectional maps will send informations from primary to secondary and from secondary to primary gadget.

Let gadgets exchanging datas, an ICMaP must be added to own list. - without writing a single line of source !

## 1.20 Programminghelp

The programminghelp is in a very early phase of development. From this window you can call the prepared Function-macros, paramete them and include them to your own sourcecode.

1. Choose the function by clicking
2. A window is opened, where you can choose the parameters
3. Press the button "Add function to your own sourcecode"
4. Two windows, the texteditor and the Input message handle are opened (4a),  
if the texteditor wasn't open itself (4b).
  - 4a. In the window Input message handling you must choose the message, in which the function should be added. This could be also a non-existing function.
  - 4b. When the texteditor is open, the function is added at the position of the Cursors.

When a function is choosen, HELP can be pressed to start the guide on that side, which describes the specific macrofunction.

## 1.21 Input message handle

This window is divided in two lists. The list on the left side consists of many possible appearing messages.

The single entries of the list means:

GadgetUp	- when a gadget was clicked
GadgetHelp	- when the pointer is over a gadget
MenuPick	- when a entry is picked from the menu
MenuHelp	- when the pointer is over a menuitem
CloseWindow	- when the close-symbol is clicked
Iconify	- when the iconify-symbol is clicked
Uniconify	- when the icon was double clicked
ActiveWindow	- when the window was activated
InactiveWindow	- when the window is gone in state "inactiv"
RawKey	- when a key was pressed
VanillaKey	- when a key was pressed (menu-shortcut)
MouseButtons	- when a mousebutton was pressed
MouseMove	- when the mouse was moved
NewSize	- when the size of a window was changed
ChangeWindow	- when the size or position of a window was changed
DisposedWindow	- when a window has done an action
IntuiTick	- all 1/10-sec a message
ShowWindow	- when a window was opened
Startup	- at program-start (by initialization)
Shutdown	- at program-finish

For detailed description of messages of the system, please consult the AutoDocs of the AmigaDeveloperCD!

If you choose one of these enties, the corresponding selection-possibilities to specify the wanted Gadget/Menu/Window-objects appears in the right list. A doubleclick on the entry in the right list generates a new function (procedure) in the texteditor, in which the own sourcecode could be included. That could be e.g. the prepared Function-macros or the systemfunctions of AmigaOS.

## 1.22 predefined variables

Name	Kind	Contains
os35	(BOOL)	= TRUE, when OS 3.5 is installed
terminated	(BOOL)	= TRUE, when program should be quitted
vinfo	(void *)	= GetVisualInfo()
Catalog	(struct Catalog *)	= OpenCatalogA()
Menu	(struct Menu *)	= CreateMenus()
AppPort	(struct MsgPort *)	MsgPort for windows and its evaluation
Screen1	(struct Screen *)	= LockPubScreen()

Variable "Screen1" has been changed for 4.0 from variable "Screen"

## 1.23 Macrofunctions

In Emperor there are some macrofunctions, which overtakes complicated operations with an easy call and the corresponding parameters. These macros are to call from the own sourcecode.

```

Windowhandling
    Emperor_CloseWindow
    Emperor_OpenWindow
    Emperor_IconifyWindow
    Emperor_UniconifyWindow

Windowattributes
    Emperor_ChangeWindowPosition
    Emperor_ChangeWindowSize
    Emperor_SetWindowBusyPointer
    Emperor_SetWindowTitle
    Emperor_SetScreenTitle

Gadgetattributes
    Emperor_GetGadgetAttr
    Emperor_GetGadgetDisabledAttr
    Emperor_GetGadgetReadOnlyAttr
    Emperor_SetGadgetAttr
    Emperor_SetGadgetDisabledAttr
    Emperor_SetGadgetReadOnlyAttr

Menuattributes
    Emperor_GetMenuItemAttr
    Emperor_SetMenuItemAttr

Misc
    Emperor_ActivateGadget
    Emperor_RethinkLayout
    Emperor_RefreshGadget
    Emperor_QuitFunc
    stringlength()
    stringtoint()
    inttostring()

```

## 1.24 Function Emperor\_CloseWindow

Function  
Macrofunction for closing windows

Synopsis  
`Emperor_CloseWindow(WindowObject)`

```
void Emperor_CloseWindow(Object *);
```

Action  
This function does some tasks for the programmer.  
1. deleting created Menu

2. closing window
3. resetting variable "Window"

#### Input

WindowObject - This must be the name of the windowobject.  
 You can paramete it in the project-manager, when you are  
 creating a new window. Template is:  
 Object \*WindowObjX; X is the number of the window;  
 e.g. Object \*WindowObj1, \*WindowObj2...;

#### Example

```
/* Closing of windows must be done by yourself, */
/* because that actions of freeing memory can't done by the maincode. */
/* If the applicationwindow is closed, Emperor isn't closing the program ! */

void Emperor_Chooser1GadgetUpEvent(void)
{
    /*
    ... pre-sourcecode ...
    ...
    ...
    ...
    */

    Emperor_CloseWindow(WindowObj1);

    /*
    ...
    ...
    ...
    ... past-sourcecode ...
    */

    Window1 = Emperor_OpenWindow(WindowObj1);

    /*
    ...
    ...
    ...
    ... sourcecode ...
    */
}
```

See also

Emperor\_OpenWindow()

## 1.25 Function Emperor\_OpenWindow

#### Function

Macrofunction for opening windows

#### Synopsis

Window = Emperor\_OpenWindow(WindowObject)

```
struct Window *Emperor_OpenWindow(Object *);
```

---

### Action

This function does some tasks for the programmer.

1. Opening of the window
2. Creating the menu

### Input

WindowObject - This must be the name of the windowobject.

You can paramete it in the project-manager, when you are creating a new window. Template is:

Object \*WindowObjX; X is the number of the window;  
e.g. Object \*WindowObj1, \*WindowObj2...;

### Result

is a variable of the type "struct Window \*". It's a pointer to the windows- ↵  
structure.

This should be that variable, which name you typed at creating the window ↵  
in the

project-manager. If you haven't gave a name for the window, it's a ↵  
variable like this:

Object \*WindowX; X is the number of the window;  
e.g. struct Window \*Window1, \*Window2...;

### Example

```
void Emperor_Button1GadgetUpEvent(void)
{
    /*
    ... pre-sourcecode ...
    ...
    ...
    ...
    */

    Emperor_CloseWindow(WindowObj1);

    /*
    ...
    ...
    ...
    ... pre-sourcecode ...
    */

    Window1 = Emperor_OpenWindow(WindowObj1);

    /*
    ...
    ...
    ...
    ... past-sourcecode ...
    */
}
```

See also

Emperor\_CloseWindow()

## 1.26 Function Emperor\_IconifyWindow

Function

Macrofunction for iconify of windows

Synopsis

Emperor\_IconifyWindow(WindowObject)

```
void Emperor_IconifyWindow(Object *);
```

Action

This function does some tasks for the programmer.

1. deleting created Menu
2. closing window
3. resetting variable "Window"

Input

WindowObject - This must be the name of the windowobject.

You can paramete it in the project-manager, when you are creating a new window. Template is:

Object \*WindowObjX; X is the number of the window;  
e.g. Object \*WindowObj1, \*WindowObj2...;

Example

```
void Emperor_Checkbox1GadgetUpEvent(void)
{
    /*
    ... some sourcecode ...
    ...
    ...
    ...
    */

    Emperor_IconifyWindow(WindowObj1);

    /*
    ...
    ...
    ...
    ... further sourcecode ...
    */

    Window = Emperor_UniconifyWindow(WindowObj1);

    /*
    ...
    ...
    ...
    ... further sourcecode ...
    */
}
```

see also

Emperor\_UniconifyWindow()

## 1.27 Function Emperor\_UniconifyWindow

### Function

Macrofunction for uniconify of windows

### Synopsis

```
Window = Emperor_UniconifyWindow(WindowObject)
```

```
struct Window *Emperor_UniconifyWindow(Object *);
```

### Action

This function does some tasks for the programmer.

1. Opening of the window
2. Creating the menu

### Input

WindowObject - This must be the name of the windowobject.

You can paramete it in the project-manager, when you are creating a new window. Template is:

Object \*WindowObjX; X is the number of the window;  
e.g. Object \*WindowObj1, \*WindowObj2...;

### Result

is a variable of the type "struct Window \*". It's a pointer to the windows- ←  
structure.

This should be that variable, which name you typed at creating the window ←  
in the  
project-manager. If you haven't gave a name for the window, it's a ←  
variable like this:

Object \*WindowX; X is the number of the window;  
e.g. struct Window \*Window1, \*Window2...;

### Example

```
void Emperor_Checkbox1GadgetUpEvent(void)
{
    /*
    ... some sourcecode ...
    ...
    ...
    ...
    */

    Emperor_IconifyWindow(WindowObj1);

    /*
    ...
    ...
    ...
    ... further sourcecode ...
    */

    Window = Emperor_UniconifyWindow(WindowObj1);

    /*
    ...
    ...
    ...
    ...
    */
```

```
...
... further sourcecode ...
*/
}
```

see also

`Emperor_IconifyWindow()`

## 1.28 Function `Emperor_ChangeWindowPosition`

Function

Macrofunction for setting the window-attributes.

Synopsis

`Emperor_ChangeWindowPosition(Window, Left, Top)`

```
void Emperor_ChangeWindowPosition(struct Window *, WORD, WORD);
```

Action

This macro is using the `ChangeWindowBox()`-Function of the Intuition-Library.  
The position of the windows is modified.

Input

Window - Window to modify. Example: `Window1`

Left - Left edge of the window, relative to the left screenborder.

Top - Top edge of the window, relative to the top screenborder.

Example

```
void Emperor_WindowPositionMenuEvent(void)
{
    /*
    ... some sourcecode ...
    ...
    ...
    ... further sourcecode ...
    */

    Emperor_ChangeWindowPosition(Window, 100, 200);

    /*
    ...
    ...
    ... further sourcecode ...
    */
}
```

see also

`Emperor_ChangeWindowSize()`

## 1.29 Function `Emperor_ChangeWindowSize`

#### Function

Macrofunction for setting of window-attributes.

#### Synopsis

Emperor\_ChangeWindowSize(Window, Width, Height)

```
void Emperor_ChangeWindowSize(struct Window *, WORD, WORD);
```

#### Action

This macro is using the ChangeWindowBox()-Function of the Intuition-Library.  
The position of the windows is modified.

#### Input

Window - Window to modify. Example: Window1

Width - Width of the window.

Height - Height of the window.

#### Example

```
void Emperor_WindowSizeMenuEvent(void)
{
    /*
    ... some sourcecode ...
    ...
    ...
    */

    Emperor_ChangeWindowSize(Window, 300, 400);

    /*
    ...
    ...
    ... further sourcecode ...
    */
}
```

see also

Emperor\_ChangeWindowPosition()

## 1.30 Function Emperor\_SetWindowBusyPointer

#### Function

Macrofunction for setting of window-attributes.

#### Synopsis

Emperor\_SetWindowBusyPointer(Window, on/off)

```
void Emperor_Window(struct Window *, BOOL);
```

#### Action

This macro is using the SetWindowPointer()-Function of the Intuition-Library.  
The pointer is toggled to "Busy"-mode.

---

#### Input

Window - Window to modify. Example: Window1  
on/off - switching the BusyPointer on ("TRUE") and off ("FALSE").

#### Example

```
void Emperor_WindowBusyPointerMenuEvent(void)
{
    /*
    ... some sourcecode ...
    ...
    ...
    */

    Emperor_SetWindowBusyPointer(Window);

    /*
    ...
    ...
    ...
    further sourcecode ...
    */
}
```

## 1.31 Function Emperor\_SetWindowTitle

#### Function

Macrofunction for setting of window-attributes.

#### Synopsis

```
Emperor_SetWindowTitle(Window, windowtitle)
```

```
void Emperor_SetWindowTitle(struct Window *, STRPTR);
```

#### Action

This macro is using the SetWindowTitles()-Function of the Intuition-Library.  
The titel of the window is modified.

#### Input

Window - Window to modify. Example: Window1  
windowtitle - new titel for the window

#### Example

```
void Emperor_WindowTitleMenuEvent(void)
{
    /*
    ... some sourcecode ...
    ...
    ...
    */

    Emperor_SetWindowTitle(Window, "new title");

    /*
```

```
...
...
...
... further sourcecode ...
*/
}
```

see also

`Emperor_SetScreenTitle()`

## 1.32 Function Emperor\_SetScreenTitle

Function

Macrofunction for setting of window-attributes.

Synopsis

`Emperor_SetScreenTitle(Window, screentitle)`

```
void Emperor_SetScreenTitle(struct Window *, STRPTR);
```

Action

This macro is using the `SetWindowTitles()`-Function of the Intuition-Library.  
The titel of the screen is modified.

Input

Window - Window to modify. Example: Window1  
screentitle - new titel for the screen

Example

```
void Emperor_WindowTitleMenuEvent(void)
{
    /*
    ... some sourcecode ...
    ...
    ...
    ... further sourcecode ...
    */

    Emperor_SetScreenTitle(Window, "new title");

    /*
    ...
    ...
    ... further sourcecode ...
    */
}
```

see also

`Emperor_SetWindowTitle()`

## 1.33 Function Emperor\_GetMenuItemAttr

---

### Function

Macrofunction for getting of menuitem-attributes.

### Synopsis

```
result = Emperor_GetMenuItemAttr(MenuObject, MenuattrID)
```

```
BOOL Emperor_GetMenuItemAttr(UBYTE, UWORD);
```

### Action

The function gives the current state of attributes of menuitems.  
You can ask, whether the item is checked, enabled and so on.

### Input

MenuObject - here you must put in the name of the menu-objects,  
which could be attributed by creating the menustrip.  
This solution is very advantageous, because the management  
of menus and its ID-numbers is relative complicable  
for beginners.

MenuFlagmask - this parameter you must set by the three following masks,  
which choose the asked attribut:

- \* ITEMENABLED (whether the item is chooseable)
- \* MENTOGGLED (for MutualExclude or Checkit;  
asks, whether the item is toggled)
- \* CHECKED (für MutualExclude oder Checkit;  
asks, whether the item is checked)

### Result

result = TRUE, if the chosen attribute is set, otherwise result = FALSE

### Example

```
void Emperor_ItemCheckitMenuEvent(void)
{
    BOOL boolean;

    /*
    ... some sourcecode ...
    ...
    ...
    */

    boolean = Emperor_GetMenuItemAttr(Menu_ItemCheckit, CHECKED);

    /*
    ...
    ...
    ... further sourcecode ...
    */
}
```

see also

```
Emperor_SetMenuItemAttr()
```

## 1.34 Function Emperor\_SetMenuItemAttr

### Function

Macrofunction for setting of menuitem-attributes.

### Synopsis

```
Emperor_SetMenuItemAttr(MenuObject, MenuattrID, Set/Reset)
```

```
void Emperor_SetMenuItemAttr(UBYTE, UWORD, BOOL);
```

### Action

The function sets the current state of attributes of menuitems.  
You can set, whether the item should be checked, enabled and so on.

### Input

MenuObject - here you must put in the name of the menu-objects,  
which could be attributed by creating the menustrip.  
This solution is very advantageous, because the management  
of menus and its ID-numbers is relative complicable  
for beginners.

MenuFlagmask - this parameter you must set by the three following masks,  
which choose the set attribut:

- \* ITEMENABLED (whether the item is chooseable)
- \* MENUTOGGLED (for MutualExclude or Checkit;  
set, whether the item should be toggled)
- \* CHECKED (für MutualExclude oder Checkit;  
set, whether the item should be checked)

Set/Reset - Set/Clear the chosen mask.

### Example

```
void Emperor_ItemCheckitMenuEvent(void)
{
    BOOL boolean;

    /*
    ... some sourcecode ...
    ...
    ...
    ...
    */

    Emperor_SetMenuItemAttr(Menu_ItemCheckit, ITEMENABLED, TRUE);
    Emperor_SetMenuItemAttr(Menu_ItemCheckit, CHECKED, FALSE);

    /*
    ...
    ...
    ...
    ... further sourcecode ...
    */
}
```

see also

```
Emperor_GetMenuItemAttr()
```

---

## 1.35 Function Emperor\_ActivateGadget

### Function

Macrofunction for activating a String- or CustomGadgets.

### Synopsis

```
Emperor_ActivateGadget (GadgetObject)
```

```
void Emperor_ActivateGadget (struct Gadget *);
```

### Action

When this function is called a StringGadget is activated and the Cursor is set into das Gadget, so that the user can make inputs, without clicking. This macro uses the Intuition-function ActivateGadget().

### Input

GadgetObject - Name des Gadgets, das aktiviert werden soll.

### Example

```
void Emperor_ActivateMenuEvent (void)
{
    /*
    ... some sourcecode ...
    ...
    ...
    */

    Emperor_ActivateGadget (String3);

    /*
    ...
    ...
    ... further sourcecode ...
    */
}
```

## 1.36 Function Emperor\_QuitFunc

### Function

Macrofunction for quitting the program.

### Synopsis

```
Emperor_QuitFunc ()
```

```
void Emperor_QuitFunc (void);
```

### Action

When you are calling that function, all opened libraries etc. are closed.

Bevor calling this function, you needn't close opened windows !

---

## Example

```
void Emperor_ProjectQuitMenuEvent(void)
{
    BOOL error;

    /*
    ... some sourcecode ...
    ...
    ...
    */

    if(error) Emperor_QuitFunc();

    /*

Ende

    */
}
```

## 1.37 Function Emperor\_RethinkLayout

## Function

Macrofunction for redrawing the whole layout.

## Synopsis

Emperor\_RethinkLayout(GadgetObject)

```
void Emperor_RethinkLayout(struct Gadget *);
```

## Action

This macro uses the LayoutGadget-function RethinkLayout().

When you are calling this function, the whole layout is redrawn.

But you can define from which position in your gadgetlist the refreshing should take place.

If you want to redraw all gadgets, you should put the value "GadgetX[0]", "X" represents the number of the window in your list in the project-manager

Important by complete refreshing is the "[0]" behind the name in the list.

This expression stands for the Root-Gadget (first Gadget in the list).

## Input

GadgetObject - Name of the Gadgets, from which position in the list of the Gadgets the objects should be refreshed.

## Example

```
void Emperor_RethinkMenuEvent(void)
{
    /*
    ... some sourcecode ...
    ...
    ...
    */
}
```

```

    Emperor_RethinkLayout (Button1);

    /*
    ...
    ...
    ... further sourcecode ...
    */

    Emperor_RethinkLayout (Gadget1[0]);

    /*
    ...
    ...
    ... further sourcecode ...
    */
}

see also
Emperor_RefreshGadget ()

```

## 1.38 Function Emperor\_RefreshGadget

Function  
Macrofunction for redrawing a single gadget.

Synopsis  
Emperor\_RefreshGadget (GadgetObject)

```
void Emperor_RefreshGadget (struct Gadget *);
```

Action  
This macro uses the Intuition-Function RefreshGList().  
When you are calling this function, declared gadget is redrawed.

Input  
GadgetObject - Name of the gadgets, which should be refreshed.

Example

```

void Emperor_RefreshMenuEvent (void)
{
    /*
    ... some sourcecode ...
    ...
    ...
    ... further sourcecode ...
    */

    Emperor_RefreshGadget (Button2);

    /*
    ...
    ...
    ... further sourcecode ...
    */
}

```

```

    ... further sourcecode ...
    */
}

```

see also

`Emperor_RethinkLayout()`

## 1.39 Function Emperor\_GetGadgetAttr

Function

Macrofunction for getting a Gadget-value.

Synopsis

```
result = Emperor_GetGadgetAttr(GadgetObject)
```

```
STRPTR Emperor_GetGadgetAttr(struct Gadget *);
```

Action

This macro uses the `GetAttr()`-Function of the Intuition-Library. Depending on the kind of the gadget this macro controls `GetAttr()`-Function. Without additional inputs, the flag-choice is made by this macro !

These flags are supported:

```

GA_Selected           (Button, Checkbox)
CHOOSEER_Selected
CLICKTAB_Current
FUELGAUGE_Level
GRAD_CurVal
INTEGER_Number
LISTBROWSER_Selected
PALETTE_Colour
RADIOBUTTON_Selected
SCROLLER_Top
SLIDER_Level
STRINGA_TextVal
TEXTEDITOR_Contents - Flag.

```

If you want to get other flags, then please use the real Intuition-Function.

Input

Gadget - which gadget should be get    Example: `Checkbox3`

Result

This function returns the flag (see above) of the gadget.

ATTENTION!! The return value is of type "STRPTR" (chain of characters).

The transformation of the value can be executed by calling these functions ↔  
:

```

stringtoint()
inttostring()

```

Bugs

The function returns a peculiar value in version 2.00.  
This bug is compensated since version 2.20.

Example

```
void Emperor_Checkbox3GadgetUpEvent(void)
{
    LONG number;
    STRPTR string;

    /*
    ... some sourcecode ...
    ...
    ...
    */

    number = stringtoint(Emperor_GetGadgetAttr(Fuelgauge2));

    strcpy(string, Emperor_GetGadgetAttr(String3));

    /*
    ...
    ...
    ...
    ... further sourcecode ...
    */
}
```

see also

Emperor\_SetGadgetAttr()

## 1.40 Function Emperor\_GetGadgetAttrComplex

Function

Macrofunction for getting a Gadget-value.

Synopsis

```
result = Emperor_GetGadgetAttrComplex(GadgetObject, GadgetattrID)
```

```
STRPTR Emperor_GetGadgetAttrComplex(struct Gadget *, ULONG);
```

Action

This macro uses the GetAttr()-Function of the Intuition-Library.  
In contrast to Emperor\_GetGadgetAttr() here you must  
set a concrete tag.

Input

Gadget - which gadget should be got    Example: Checkbox3

GadgetattrID - which GadgetattrID should be used    Example: ↔  
CHECKBOX\_TextPlace

Result

This function returns the value of the given tag.

ATTENTION!! The return value is of type "STRPTR" (chain of characters).

The transformation of the value can be executed by calling these functions ↵  
:

```
stringtoint()
inttostring()
```

Example

```
void Emperor_Checkbox3GadgetUpEvent(void)
{
    LONG number;
    STRPTR string;

    /*
    ... some sourcecode ...
    ...
    ...
    ...
    */

    number = stringtoint(Emperor_GetGadgetAttrComplex(Fuelgauge2, FUELGAUGE_Min));

    strcpy(string, Emperor_GetGadgetAttrComplex(Button3, GA_Text));

    /*
    ...
    ...
    ...
    ... further sourcecode ...
    */
}
```

see also

```
Emperor_SetGadgetAttrComplex()
```

## 1.41 Function Emperor\_GetGadgetDisabledAttr

Function

Macrofunction for getting the GA\_Disabled-attributes.

Synopsis

```
result = Emperor_GetGadgetDisabledAttr(GadgetObject)
```

```
BOOL Emperor_GetGadgetDisabledAttr(struct Gadget *);
```

Action

This macro uses the GetAttr()-Function of the Intuition-Library.  
The GA\_Disabled-Flag (whether the gadget is chooseable) is set  
and returned.

Input

Gadget - which gadget should be got      Example: Checkbox3

Result

The function returns the GA\_Disabled-Flag of the corresponding gadget.

Example

```
void Emperor_Integer1GadgetUpEvent(void)
{
    BOOL result;
    /*
    ... some sourcecode ...
    ...
    ...
    */

    result = Emperor_GetGadgetDisabledAttr(Button1);

    /*
    ...
    ...
    ...
    further sourcecode ...
    */
}
```

see also

```
Emperor_SetGadgetDisabledAttr()
```

## 1.42 Function Emperor\_GetGadgetReadOnlyAttr

Function

Macrofunction for getting the GA\_ReadOnly-attributes.

Synopsis

```
result = Emperor_GetGadgetReadOnlyAttr(GadgetObject)
```

```
BOOL Emperor_GetGadgetReadOnlyAttr(struct Gadget *);
```

Action

This macro uses the GetAttr()-Function of the Intuition-Library.

The GA\_ReadOnly-Flag (whether the gadget is only for reading) is set and returned.

Input

Gadget - which gadget should be got    Example: Checkbox3

Result

The function returns the GA\_ReadOnly-Flag of the corresponding gadget.

Example

```
void Emperor_Integer1GadgetUpEvent(void)
{
    BOOL result;
    /*
    ... some sourcecode ...
    ...
    ...
    */
}
```

```

    result = Emperor_GetGadgetReadOnlyAttr(Button1);

    /*
    ...
    ...
    ... further sourcecode ...
    */
}

```

see also

```
Emperor_SetGadgetReadOnlyAttr()
```

## 1.43 Function Emperor\_SetGadgetAttr

Function

Macrofunction for setting a Gadget-value.

Synopsis

```
Emperor_SetGadgetAttr(GadgetObject, Value)
```

```
void Emperor_SetGadgetAttr(struct Gadget *, STRPTR);
```

Action

This macro uses the SetGadgetAttr()-Function of the Intuition-Library.

Depending on the kind of the gadget this macro controls SetGadgetAttr()-Function ↔

Without additional inputs, the flag-choice is made by this macro !

These flags are supported:

```

GA_Selected           (Button, Checkbox)
CHOOSEER_Selected
CLICKTAB_Current
FUELGAUGE_Level
GRAD_CurVal
INTEGER_Number
LISTBROWSER_Selected
PALETTE_Colour
RADIOBUTTON_Selected
SCROLLER_Top
SLIDER_Level
STRINGA_TextVal
TEXTEDITOR_Contents - Flag.

```

If you want to set other flags, then please use the real Intuition-Function.

Input

Gadget - which gadget should be set    Example: Checkbox3

Value - Value, how the flag should be modified.

ATTENTION!! The value must be from type "STRPTR" (chain of characters).

The converting can be done by using these functions:

```
stringtoint()
```

```
inttostring()
```

Example

```
void Emperor_Checkbox3GadgetUpEvent(void)
{
    /*
    ... some sourcecode ...
    ...
    ...
    */

    Emperor_SetGadgetAttr(Fuelgauge2, "23");

    Emperor_SetGadgetAttr(String3, "String");

    /*
    ...
    ...
    ...
    ... further sourcecode ...
    */
}
```

see also

```
Emperor_GetGadgetAttr()
```

## 1.44 Function Emperor\_SetGadgetAttrComplex

Function

Macrofunction for setting a Gadget-value.

Synopsis

```
Emperor_SetGadgetAttrComplex(GadgetObject, GadgetattrID, Value)
```

```
void Emperor_SetGadgetAttrComplex(struct Gadget *, ULONG, STRPTR);
```

Action

This macro uses the SetGadgetAttr()-Function of the Intuition-Library.  
In contrast to Emperor\_SetGadgetAttr() here you must  
set a concrete tag.

Input

Gadget - which gadget should be set    Example: Checkbox3

GadgetattrID - which GadgetattrID should be used    Example:  $\leftarrow$   
CHECKBOX\_TextPlace

Value - Value, how the flag should be modified.

ATTENTION!! The value must be from type "STRPTR" (chain of characters).  
The converting can be done by using these functions:

```
stringtoint()
inttostring()
```

Example

```
void Emperor_Checkbox3GadgetUpEvent(void)
{
    /*
    ... some sourcecode ...
    ...
    ...
    */

    Emperor_SetGadgetAttrComplex(Fuelgauge2, FUELGAUGE_Min, "23");

    Emperor_SetGadgetAttrComplex(String3, STRINGA_MaxChars, "10");

    Emperor_SetGadgetAttrComplex(Listbrowser1, LISTBROWSER_MinVisible, "20");

    Emperor_SetGadgetAttrComplex(Listbrowser1, LISTBROWSER_Labels, (STRPTR) & ↵
        Listbrowser_NewListArray);
    /*****
    /* This isn't a bug in the guide.... */
    /* When changing Labels, you must pay attention */
    /* that you cast the argument with (STRPTR) */
    /* and give the address (&!! to the function */
    /*****/

    /*
    ...
    ...
    ...
    ... further sourcecode ...
    */
}
```

see also

```
Emperor_GetGadgetAttrComplex()
```

## 1.45 Function Emperor\_SetGadgetDisabledAttr

Function

Macrofunction for setting the GA\_Disabled-attributes.

Synopsis

```
Emperor_SetGadgetDisabledAttr(GadgetObject, disable/enable)
```

```
void Emperor_SetGadgetDisabledAttr(struct Gadget *, BOOL);
```

Action

This macro uses the SetGadgetAttr()-Function of the Intuition-Library.  
The GA\_Disabled-Flag (whether the gadget is chooseable) is set.

Input

Gadget	- which gadget should be set	Example: Checkbox3
dis/enable	- TRUE to disable, FALSE to enable flag.	

Example

```
void Emperor_Integer1GadgetUpEvent(void)
{
    /*
    ... some sourcecode ...
    ...
    ...
    */

    Emperor_SetGadgetDisabledAttr(Button1, TRUE);
    Emperor_SetGadgetDisabledAttr(Checkbox3, FALSE);

    /*
    ...
    ...
    ...
    ... further sourcecode ...
    */
}
```

see also

```
Emperor_GetGadgetDisabledAttr()
```

## 1.46 Function Emperor\_SetGadgetReadOnlyAttr

Function

Macrofunction for setting the GA\_ReadOnly-attributes.

Synopsis

```
Emperor_SetGadgetReadOnlyAttr(GadgetObject, on/off)
```

```
void Emperor_SetGadgetReadOnlyAttr(struct Gadget *, BOOL);
```

Action

This macro uses the SetGadgetAttr()-Function of the Intuition-Library.  
The GA\_ReadOnly-Flag (whether the gadget is chooseable) is set.

Input

Gadget	- which gadget should be set	Example: Checkbox3
on/off	- TRUE to make gadget readonly, FALSE to make gadget choosable.	

Example

```
void Emperor_Integer1GadgetUpEvent(void)
{
    /*
    ... some sourcecode ...
    ...
    ...
    */

    Emperor_SetGadgetReadOnlyAttr(Button1, FALSE);
    Emperor_SetGadgetReadOnlyAttr(Checkbox3, TRUE);
}
```

```
/*
...
...
... further sourcecode ...
*/
}
```

see also

```
Emperor_GetGadgetReadOnlyAttr()
```

## 1.47 Function stringtoint

Function

This function converts characters (STRPTR = chains of characters) into an integer number (LONG = 32-bitnumber).

Synopsis

```
result = stringtoint(string)
```

```
LONG stringtoint(STRPTR);
```

Action

It is for transformations of the results of functions like:

```
Emperor_GetGadgetAttr
Emperor_SetGadgetAttr
```

Input

string - characters to convert

Example

```
void Emperor_Cut_MenuEvent(void)
{
    LONG number;
    STRPTR string = "-123456789";
```

```
/*
... some sourcecode ...
...
...
... further sourcecode ...
*/
```

```
number = stringtoint(string); /* !! number = -123456789 !! */
```

```
/*
...
...
... further sourcecode ...
*/
}
```

---

see also  
`inttostring()`

## 1.48 Function `inttostring`

### Function

This function converts an integer number (LONG = 32-bitnumber) into characters.

### Synopsis

```
result = inttostring(number)
```

```
STRPTR inttostring(LONG);
```

### Action

This function converts a number of the area from -999999999 to 999999999 into characters (= STRPTR).  
It is for transformations of the results of functions like:

```
Emperor_GetGadgetAttr  
Emperor_SetGadgetAttr
```

### Input

number - number to transform

### Example

```
void Emperor_Cut_MenuEvent(void)  
{  
    STRPTR string;  
  
    /*  
    ... some sourcecode ...  
    ...  
    ...  
    ...  
    */  
  
    strcpy(string, (char *) inttostring(-123456789));  
  
    /*  
    ...  
    ...  
    ...  
    ... further sourcecode ...  
    */  
}
```

see also  
`stringtoint()`

## 1.49 Function `stringlength`

---

#### Function

This function returns the length of a string.

#### Synopsis

```
result = stringlength(string)
```

```
ULONG stringlength(STRPTR);
```

#### Input

string - string, you want to get the length from

#### Beispiel

```
void Emperor_Cut_MenuEvent(void)
{
    ULONG result;

    /*
    ... some sourcecode ...
    ...
    ...
    */

    result = stringlength("Hello"); /* result = 5 */

    /*
    ...
    ...
    ... further sourcecode ...
    */
}
```

## 1.50 History

2.0 - Initial release

2.1 - AmigaOS 2.0 Gadget support !!

(GadTools-Gadgets are implemented now !)

- reached to shrink programsize by coding the window-stuff by myself instead of making it by ReActor

2.2 - Oops..The program crashed, when a clicktab was in a test-window.

- speed optimizations (e.g. at startup, shutdown etc.)
- defined catalog-creation
- C++-Code support
- added the macro "Preferences-Menu" for Prefs-Programs
- better & easier generated C-sourcecode
- minor bugfixes

2.3 - prefs-editor rewritten

- chooseable procestortype for generated StormC-project
  - modified programming-help
  - some functionmacros rewritten
-

- minor bugfixes
- 3.0
- gadgets-test routine rewritten
  - fixed bugs by showing tapedeck & colorwheel gadgets
  - fixed problems with clicktabs & page gadgets (see "clicktabsexample")
  - added function "Shutdown()" (called, when program quits)
  - added progress-bar while saving
  - added macrofunctions Emperor\_Set/GetGadgetAttrComplex()
- 3.1
- fixed a bug when generating source for window with colorwheel
  - source is now \*more\* SAS/C friendly
  - fixed a silly bug when copy a function from programming help into ↵  
texteditor
  - added online help for macrofunctions (see chapter Programming Help for ↵  
details)
  - added ARexx-connection to StormC (Scripts - Dir)
  - added arguments by starting StormC (e.g. GoldED support for StormC)
  - added loading picture
- 3.2
- added parametry of several image-gadgets (bevel, bitmap, glyph)  
(see "ClicktabExample" for details)
  - fixed a bug with slider and getfile-gadgets
  - fixed problem with requester-body-text ["\n" (Newline) will be recognized ↵  
now]
  - rewritten routine for adding, moving (up and down) and removing of objects ↵  
in a list
- 3.5
- showing of scroller & slider-gadgets fixed
  - layout-gadget (bevelstyle & -state) doesn't switch back to previous setted  
values anymore
  - added support of finnish catalog
  - added program-environment for configuration of version, program name,  
stack, mem and many more
  - deleting windows is possible now
  - when pages were added to gadgetlist, its generation was incorrect
  - added gadgethelp support - just connect a gadget to configure ! (program- ↵  
environment)
  - added function <Menu-Edit-Clearlist> to reset current list of objects
  - label's Locale-connection works correct now
  - gadgethelp for nearly every gadget
  - added chooseable HookType for string-gadgets
  - added some initial-configurations for texteditor-gadgets
  - moving Locale-entrys is possible now
  - editing the includelist is also possible now
  - added new startup-picture (thanks to Janne Peräaho)
- 3.6
- Gradientsliders horiz/vert orientation works correct now
  - some global variables problems fixed
  - problems with node-array fixed
  - special positions/sizes for OS 2.0 windows work now
  - chooseable smallfont for large lists
  - fixed crash when moving a test-window
  - modifyable library-integration for your project
  - fixed bug in installer-script (doesn't notice, if an OS higher than 3.5 was ↵  
installed)
  - added interconnection maps
  - generating source based on AmigaOS 2.0, AmigaOS 3.5+ or mixed code
-

- 4.0 - DONE ! the whole code is rewritten and structured ! only 4 MBytes of systemmemory is needed !
- the programflow maybe changed a bit to a better, because also all low-level functions were rewritten
  - look also changed a bit
  - menu is more complex and professional
  - expanded GadTools-handling (with RightMouseButton)
- 4.1 - added french catalog (written by Olivier Martin -- thank you !)
- added translations for predefined menus and requests
  - added attributes for texteditor-, speedbar- and space-gadget
  - new special requester-kind (in Reaction-style)
  - ARexx-script changed/fixed problem while transferring projects
  - removed fix-positions for attribute-windows
  - fixed problem with preview-menu (no wrong additional menuitems anymore)
  - added array for speedbar buttons
  - fixed problems with arrays
  - enlarged edges around GadTools-gadgets for resizing
  - path to includes and libs configurable
  - layout of mainwindow changed (no overlapping gadgets anymore)
  - added requester, if choosing menuitem "New"
  - fixed a bug in menuitem "New"
  - fixed a bug in menuitem "Delete"
  - bugs within generated code were fixed
  - chooseable generation of macrofunctions
  - generated code much smaller while leaving out unused macrofunctions and restructuring it (ca. 40 % smaller)
  - chooseable initial opened window for your project
  - fixed bug in source, when there was just one gadget in a list
  - added possibility to create a directory containing the current project
  - bug while opening old projects containing a slider fixed
  - chooseable locale-datafile creation
  - chooseable storm-projectfile creation
  - texteditor changes are recognized before saving
  - saving of strings containing a linefeed ('\n') is no problem anymore
  - opened amigaguide.library because of some initialization-bugs

## 1.51 FAQ

Question: When I change the size of a GadTools-gadget, the border, which marks the activation, isn't (or just a bit) visible.

Answer: Probably you are using a program like VisualPrefs, which patches the visualization of GadTools-gadgets. If you deactivate it, it should work.  
Or you may change your border-sizes of a "String"- and "Integer"-gadgets down to normalsize, which does have the same effect.

Question: Emperor doesn't run on my system !

Answer:        There are a few things, you must pay attention on:

- \* AmigaOS 3.5 installed ?
- \* 4 MBytes of systemmemory must be free on startup
- \* assign on <PATH>:Emperor
- \* RexxMast already started ?
- \* stack at >50000 bytes ?
  - under CLI:    type 'stack 50000'
  - under WB:    in iconinformations choose a stack of >50000 bytes
  - under DOpus: in menu 'Settings-Environment' choose point 'CLI- ↵  
                  Launching' and  
                  change stack !

To every occurred error at startup should appear a request...

Question:     In the preview-window labels and bitmaps were not shown.

Answer:        That's an very old bug in Emperor and I'm still working on that ↵  
                  problem !

## 1.52 known bugs & future plans

known bugs of the program:

1. Printing the objecttable is deactivated.  
This bug shouldn't have an influence to the capability of functions of ↵  
the program.
2. The attribut-window "Datebrowser" is much too wide (when choosing a ↵  
nonproportional  
screenfont) ! (but why .... ?)
3. Saving clips has no function. (deactivated)
4. Text- or lineformatting is not implemented yet. (deactivated)
5. Labels and bitmaps aren't shown in the preview-window

future plans:

1. Screen-category
2. Undo/Redo action for lists
3. Checking items of the lists
4. Insertmode isn't supported by the ReAction-Texteditor-Gadget

## 1.53 Author

---

If you have some suggestions for improving or correcting the program,  
please write or mail me your facts to:

The address of the author:

Matthias Gietzelt  
Ringstraße 41  
19069 Hundorf  
Germany

eMail:  
shamane@exmail.de

## 1.54 Expression of thanks

I want to thank very, very much Janne Peräaho for his nice startup-picture he made ↵  
.  
Also for his bugreports, to correct existing errors in Emperor.  
- Thanks Janne !

Olivier Martin for his bugreports and his highend-example project "Tiger". You can  
find this example in the directory "projects" of Emperor. It is a very nice tool ↵  
to show  
the developing of "virtual life" on Amiga - check this out !  
- Thanks Olivier !

Also I want to say thanks very much to all the bugreporters for the thorough use ↵  
of Emperor !!

---