GUI

COLLABORATORS			
	TITLE : GUI		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		November 28, 2024	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

GUI

1.1 GUI.guide

```
                                GUIEnvironment

                          Graphical User Interface Guide

 ========================================================================
```


```
                            © 1994   Carsten Ziegeler
                                     Augustin-Wibbelt-Str.7
                                     D-33106 Paderborn
                                     Germany
```


```
 ========================================================================

   Fonts

   Screens

   Windows


   GUI Handling

   Font adaptive GUI

   Changing attributes / appearance

   Programming guidelines

   The GUIInfo structure

   The GUI tags

   GUIEnvironment error codes


SEE ALSO
     Gadgets
     Menu
```

1.2 Fonts

< Not yet written - wait for the next release >

1.3 Screens

< Not yet written - wait for the next release >

1.4 Windows

It is currently not possible to use gimmeZeroZero windows with GUIEnv !

GUIEnvironment takes the width and height for the window as inner width
and inner height to make the window fit to every font.
If you need the specified width and height instead, use the GEW_OuterSize
tag with any value !

< Not yet completed - wait for the next release >

1.5 GUI Handling

The first step your applications has to do, is to open a window (and
perhaps before a screen and fonts).
After this you need the very important GUIInfo structure, GUIEnvironment
is based on.
You can get this with the CreateGUIInfo function. It is possible to
give some tags, which will fill in some entries of the GUIInfo structure.

If the creation was successfull, you now can define your GUI with gadgets
and a menu strip. If you have defined everything, the GUI must be drawn
using DrawGUI !
After this the GUI is displayed in the window and you have to do the
message loop to get the user action.

Exiting your application, you have to free the GUIInfo structure with
the FreeGUIInfo function ! After freeing it, you can close your window.

SEE ALSO
The GUI tags

1.6 Font adaptive GUIs

When creating the GUIInfo, use the GUI_CreationFont tag to specify the
font the GUI originally was designed for. The adapting can now be done
automatically by GUIEnvironment. It resizes and repositions each gadget
and text !
When the GUI_CreationFont tag is specified, GUIEnvironment usually changes
the window sizes to make the GUI fit for the new (bigger) font ! The min
and max values of the window are also changed !
If you don't want this, use the GUI_PreserveWindow tag ! If the tag data
is GUI_PWFull the window will chainged unchanged and also the min and
max values will. Using GUI_PWSize will only change the min and max values
and GUI_PWMinMax will only change the window size !

Remember to use the GUI_CreationWidth and GUI_CreationHeight tags !
If the window didn't fit on the screen the creation font is again used
as the text font, so the GUI will always appear, but your application
will not get any message that the used font has changed !

The window is also repositioned to fit on the screen if necessary.

Each time the GUI_CreationWidth/GUI_CreationHeight tags are changed,
the window will be resized, so don't forget to use GUI_PreserveWindow
each time you need it !

SEE ALSO
     The GUI tags

1.7 Changing attributes / appearance

The GUIInfo structure contains a lot of information which can be
changed by the application. But because the structure is read only
you can't simply change any entry of the structure.
For this reason the ChangeGUI function was designed. You can pass
it a lot of tags, where each tags stands for an entry of the
structure.

By using ChangeGUI it is also possible to change the appearance
of the GUI. This includes removing the menu strip, or removing
the gadgets.

SEE ALSO
     The GUI tags

1.8 Programming Guidelines

The GUIInfo structure which nearly all procedures require, contains
all important data.
Furthermore, the real structure contains a lot of hidden data at the end,
so create and handle this structure only with the functions provided by
GUIEnvironment.
In further versions this structure will contain additional data.

Don’t use any functions of the gadtools.library anymore. GUIEnvironment replaces the whole gadtools.library (except the message filter functions), so there is no need to use gadtools.library ! (It is still possible to use the message filter functions, if you take some extra care !)

Except the freeing functions of GUIEnvironment, there is no check of the parameters. So you MUST handle the problem if the GUIInfo structure couldn’t be created ! Passing NULL to any of the GUI handling procedures will propably crash the task (or even worse!).
Neither, the functions do any gadget number/ ID check, so be sure that you only pass valid values !

Remember that GUIEnvironment uses the UserData entries of the gadgets and the menu items to store some own important information !
This means you can’t use it to store some own user specific data in this field or some VERY STRANGE THINGS will happen. (Usually then nothing more will happen, because the computer has hung itself !)
But don’t worry, you stil have the possibility to store own user data. See the appropriate chapters about gadgets and menu items.

If you demand something (screens, windows, fonts etc) with GUIEnvironment you have to free it with GUIEnvironment again !

1.9 The GUIInfo structure

The GUIInfo structure is the most important structure for GUIEnvironment. It contains all necessary information about the GUI, e.g. the gadgets or the menu layout.
In addition all important structures dealing with GUIs are stored there, e.g. DrawInfo, VisualInfo or even Locale.
Because GUIEnvironment has its own message handling routines, GUIInfo also contains some information about the incomming messages.

Now follows the whole structure:

```
struct GUIInfo {

  struct Window *window;

         This is the window for which the GUI is defined

  struct Screen *screen;

         Points to the window’s screen

  APTR   visualInfo;

         Pointer to the VisualInfo of the screen

  struct DrawInfo *drawInfo;

         Pointer to the DrawInfo of the screen
```

```
struct Locale *localeInfo;

       Pointer to the current Locale structure. This is only available
       if the locale.library is installed !

struct TextAttr *menuFont;

       The font used to layout the menu.

WORD creationWidth, creationHeight;

       The size of the GUI the layout was originally designed for

struct MsgPort *msgPort;

       The port to handle all intuition messages

struct IntuiMessage *intuiMsg;

       Copy of the arrived intuition message

ULONG msgClass;

       The message class

WORD  msgCode;

       The message code

UBYTE msgBoolCode;

       For gadtools checkbox gadgets: The state of the gadget if
       a message for this gadget arrives

char  msgCharCode;

       The character of an vanilla key message

struct Gadget *msgGadget;

       Points to the event gadget if an intuition message arrived

struct MenuItem *msgItemAdr;

       Points to the event menu item if an intuition message arrived

WORD msgGadNbr;

       Gadget number / ID

WORD msgMenuNum, msgItemNum, msgSubNum;

       The appropriate numbers of the menu item.

APTR userData;

       Pointer for own user data
```

```
  APTR compilerReg;

         Usually contains the global data pointer for compilers which
         use the A4 register to store this. If a hook functions is
         called, the A4 register is set to this value !

  STRPTR gadgetGuide, menuGuide;

         The filenames of the help files to use with the GUIEnvironment
         help functions

  struct Catalog *catalogInfo;

         Points to a given catalog, only if the locale.library is
         installed.

  LONG gadgetCatalogString, menuCatalogString;

         The offsets for the next string taken out of the catalog
         for the next gadget resp menu item

  struct Hook *vanKeyHook, *handleMsgHook, *refreshHook;

         The hook functions

  APTR   hookInterface;

         This function is set in the hook.entry field. It is required
         for compilers which can't handle register parameters. Then
         this is a pointer to a function which pushes the registers
         onto the stack in the desired order.

  struct TextAttr *creationFont;

         The font used for designing the GUI.

  struct TextAttr *textFont;

         Default font for texts and gadgets. Is set to the windows font.
};

BUT NOTICE: The whole structure is READ ONLY !! It is possible to set
the entries when creating the structure. From now on you have to use
the changing routines to write to the entries !


SEE ALSO:
     Creating the GUIInfo structure
     Changing the GUIInfo structure
```

1.10 The GUI tags

```
After each tag are some characters which specify when this tag can
be used:
```

```
  C : Allowed when creating the GUIInfo structure
  S : Allowed for changing

In brackets follows the default value for creating!

The first list contains the tags which are for some values in the
GUIInfo structure and the second list consists of some action tags
which will change the appearance of the GUI !

Standard tags
-------------

 GUI_TextFont               C              (window's font)

  The standard TextAttr structure for the gadgets and texts. (It is
  also possible to give each gadget a different font !)

 GUI_MenuFont               C              (screen's font)

  The font for layouting the menu.

 GUI_VanKeyAHook            C S            (no hook function)

  The vanilla key message hook for gadget key equivalents.

 GUI_CreateError            C              (NULL)

  The data of this tag must be an address of a LONG variable, which
  containts the error code if CreateGUIInfo failt.

 GUI_UserData               C S            (NULL)

  Pointer to own user data.

 GUI_CompilerReg            C S            (value of A4 register)

  The contents of the global data register for this applications.
  If a hook function of GUIEnvironment is called the A4 register
  will get this value.

 GUI_HandleMsgAHook         C S            (no hook function)

  The message handling hook.

 GUI_GadgetGuide            C S            (no gadget help)

  Name of an AmigaGuide file which contains the help texts for the
  gadgets.

 GUI_MenuGuide              C S            (no menu help)

  Name of an AmigaGuide file which contains the help texts for the
  menu items.

 GUI_CatalogFile            C S            (no catalog file)

  Name of an catalog file for the localization.
```

```
 GUI_GadgetCatalogOffset    C S            (0)

  Number of the first gadget text inside the catalog.

 GUI_MenuCatalogOffset      C S            (0)

  Number of the first menu item text inside the catalog.

 GUI_CreationWidth          C S            (window inner width)

  The inner width of the GUI the creation values are for.

 GUI_CreationHeight         C S            (window inner height)

  The inner height of the GUI the creation values are for.

 GUI_CreationFont           C S            (window font)

  The font the GUI was designed with

 GUI_RefreshAHook           C S            (no hook function)

  The refreshing message hook.

 GUI_MsgPort                C S            (windows user port)

  The message port used for the IDCMP messages. If you change this
  port, first the old port is emptied and then the new one also !

 GUI_HookInterface          C S            (no hook interface)

  This is a interface procedure which is called for every callback hook
  to push the registers onto the stack. It is only required for compilers
  which can't handle register parameters !
  You must specify this tag BEFORE all tags which handle callback hooks.
  It is adviced to make this tag the first one, when creating the GUIInfo
  structure.

 GUI_PreserveWindow         C S

  This tag is used for adapting the GUI to a user defined font.
  Tag data:
  GUI_PWFull   : Window and min/max values won't be changed
  GUI_PWSize   : Only min/max values are changed
  GUI_PWMinMax : Only the window is changed


Action tags
-----------

 GUI_RemoveMenu             S

  Remove the menu from the window. If the tag data is TRUE all information
  about the old menu is deleted and it is not possible to add this menu
  to the window again without redefining ! If you pass FALSE as tag data,
  you can reset the menu with the appropriate functions.
```

```
GUI_RemoveGadgets          S

 Remove the gadgets from the window. If the tag data is TRUE all infor-
 mation about the old gadgets is deleted and it is not possible to add
 these gadgets to the window again without redefining ! If you pass FALSE
 as tag data, you can reset the gadgets with the appropriate functions.
 In addition the window contents is cleared.

GUI_ClearWindow            S

 Clear the window contents. This includes NOT removing the gadgets !

GUI_EmptyMsgPort           S

 Remove all outstanding messages from the message port.

GUI_DoBeep                 S

 Display the screen beep, e.g. to warn... (tag data must be set to 0)

GUI_Lock                   S

 Locks the window/GUI and displays a wait pointer. This requires the
 reqtools.library, because the LockWindow function is used for this !
 For more information refer to the ReqTools.docs.
 Tag data must be 0.

GUI_UnLock                 S

 Unlock the window/GUI locked with GUI_Lock.
 Tag data must be 0.
```

1.11 The GUIEnvironment error codes

```
If any error occurs, it is the best thing to hold the programm and
to ask the user to restart the application.
Notice that some errors will only return while testing the applica-
tion. For example the GE_GadTooManyErr or the GE_GadUnknownKind
error should never happen in a full test application !


General errors:

 GE_Done

      No error occured, go on !

 GE_MemoryErr

      Not enough memory.

 GE_WindowErr

      NULL was passed for the window pointer. GUIEnvironment needs
```

an opened window to create the GUIInfo structure !

GE_VisualInfoErr

Unable to get the VisualInfo of the screen.

GE_DrawInfoErr

Unable to get the DrawInfo of the screen.

Gadget errors:

GE_GadContextErr

The call to gadtools CreateContext failt.

GE_GadCreateErr

Unable to create the gadget. There are many reasons for this error, e.g. wrong gadget sizes/positions, unknown or not enough information about the gadget etc.

GE_GadTooManyErr

GUIEnvironment allows only 256 gadgets per GUIInfo structure !

GE_GadKeyTwiceErr

You tried to give two gadgets the same key equivalent.

GE_GadUnknownKind

Unknown gadget kind. Check the value passed to the gadget creation function.

GE_GadChainErr

This error occurs if you started the chaining function and then forgot the end chain statement or if you passed an end statement without having started a chaining !

GE_GadHookErr

Your hook function, to create this gadget, failed.

Menu errors:

GE_MenuCreateErr

The call to gadtools CreateMenus failed.

GE_MenuStripErr

The call to intuition SetMenuStrip failed.

GE_MenuLayoutErr

The call to gadtools LayoutMenus failed.

GE_MenuTooManyErr

GUIEnvironment allows only 256 menu items per GUIInfo structure.

1.12 rcs

$RCSfile: GUI.guide $

$Revision: 1.5 $
$Date: 1994/11/03 15:51:19 $

GUIEnvironment GUI Main Guide

Augustin-Wibbelt-Str.7, 33106 Paderborn, Germany