

hooks

COLLABORATORS

	<i>TITLE :</i> hooks		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY		July 24, 2024	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1	hooks	1
1.1	GUIEnvironment/Hooks guide	1
1.2	GUIEnvironment Hook Functions Guidelines	1
1.3	The vanilla key hook for gadgets' key equivalents	2
1.4	Message handling hook	2
1.5	Refreshing hook	3
1.6	Gadget event hook	3
1.7	Menu event hook	3
1.8	Gadget creation hook	4
1.9	rca	4

Chapter 1

hooks

1.1 GUIEnvironment/Hooks guide

GUIEnvironment

Callback Hooks Guide

=====

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

=====

Hook functions guidelines

Hook for gadgets' key equivalents

Message handling hook

Refreshing hook

Gadget event hook

Menu event hook

Gadget creation hook

1.2 GUIEnvironment Hook Functions Guidelines

GUIEnvironment uses for more "application interaction" the amiga callback hooks as introduced in the OS2.04+.

But the main difference is, that the tags of GUIEnvironment except NOT a Hook structure, BUT the function itself ! GUIEnvironment then creates

a Hook structure and sets the entries for the application !
 The data entry of the structure is always set to the GUIInfo structure !

Usually, the function is called directly, this means the parameters are passed using the registers A0 upto A2. As some compilers are not able to use registers directly, it is possible to define an interface which pushes these registers onto the stack and then the function is called. Specifying the GUI_HookInterface tag with the data of such an interface, causes GUIEnvironment to set the entry field of the Hook structure to this interface and the subEntry field to the function !

Another major difference between usual callback hooks and the ones used within GUIEnvironment is, that GUIEnv sets the A4 register to the entry of the GUIInfo compilerReg field !
 So it is possible to use all global data inside the hook function without setting the A4 explicitly. Of course, this works only if the compiler addresses the global data through the A4 register.

Each hook gets in A0 a pointer to an initialized Hook structure.
 Although GUIEnvironment creates the Hook structure for the application, it is still possible to change it for own needs !

1.3 The vanilla key hook for gadgets' key equivalents

The vanilla key hook is for gadgets' key equivalents which can not be handled by GUIEnvironment automatically.
 GUIEnvironment can only handle letters ('a' to 'z'). But if you need some "strange" key equivalents (e.g '+' or '-') you can define which gadget belongs to which key using this callback hook:

```
A1      : Currently unused, is set to NULL
A2      : LONG : The ASCII character code

RESULT  : LONG
```

Return GEH_KeyUnknown if the key is not a key equivalent or the number of the gadget, or even the number of the gadget plus GEH_KeyShifted if the key should be treated as a shifted one !

This hook is activated with the GUI_VanKeyAHook tag.

1.4 Message handling hook

Using the GUI_HandleMsgAHook tag, you activate a callback hook which is called for each arriving message. You can now examine this message. GUIEnvironment sets all fields of the GUIInfo structure concerning messages for you.

To examine the message use the intuitMsg field !

```
A1, A2  : Currently unused, are set to NULL
RESULT  : LONG, handled as boolean
```

Return TRUE, if GUIEnv should not work on the message anymore, otherwise FALSE.

Don't return TRUE with IDCMP_REFRESHWINDOW messages, because GUIEnv must do a lot of refreshing work which actually can't be done by the applications !

1.5 Refreshing hook

Using the GUI_RefreshAHook, you can define a callback which is called with every IDCMP_REFRESHWINDOW message. It is called after GUIEnv has done its refreshing and before the refreshing phase is ended.

A1, A2 : Currently unused, are set to NULL

RESULT : LONG , Currently unused, set this always to 0 !

1.6 Gadget event hook

Using the GEG_UpAHook and the GEG_DownAHook tag, you can define two callback hooks, which are called with the IDCMP_GADGETUP resp the IDCMP_GADGETDOWN message, if GUIEnvironment can't handle this message for the application.

A2 : Pointer to event gadget

A1 : Currently unused, is set to NULL

RESULT : LONG, handled as boolean

If you want to wait for further messages return TRUE, otherwise FALSE to exit the message-loop with WaitGUIMsg.

If you use GetGUIMsg and you returned TRUE, the msgClass field will be empty !

SEE ALSO

Gadget message handling

1.7 Menu event hook

Using the GEM_AHook tag you can specify a callback hooks which is called with every IDCMP_MENUPICK message for this menu item.

A2 : Pointer to event menu item (if possible)

A1 : Currently unused, is set to NULL

RESULT : LONG, handled as boolean

If you want to wait for further messages return TRUE, otherwise FALSE to exit the message-loop with WaitGUIMsg.

If you use GetGUIMsg and you returned TRUE, the msgClass field will be empty !

SEE ALSO

Menu message handling

1.8 Gadget creation hook

The gadget creation hook is called every time the gadget is created. This is the first time the gadget is defined and then every time the gadget is resized/repositioned or redrawn !

The hook is useful for gadtools GENERIC_KIND gadgets which need more information than given with the creation. Because every time the gadgets is resized or redrawn the gadget is created totally new, the information has to be added again and again. This is the reason for this hook:

A2 : Pointer to event gadget
A1 : Currently unused, is set to NULL

RESULT : LONG, handled as boolean.

If your creation hook has done his work, return TRUE, otherwise FALSE to stop creation !

Add the hook to the gadget with the GEG_CreationAHook tag.

1.9 rcs

\$RCSfile: Hooks.guide \$

\$Revision: 1.5 \$

\$Date: 1994/11/03 15:51:47 \$

GUIEnvironment Callback Hooks Guide

Copyright © 1994, Carsten Ziegeler

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