in

COLLABORATORS			
	TITLE : in		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		April 15, 2022	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

in

1.1 AMOSPro GUI Extension

AMOSPro GUI Extension

Welcome to the AMOSPro GUI Extension AmigaGuide.

You can read the whole manual now by selecting Browse or select one of the following:

1.2 Introduction

Introduction

The GUI extension is a powerful, easy-to-use extension for giving your programs a Graphical User Interface (GUI) in the Workbench environment. To create your GUIs, you will need the GadToolsBox editor. This can be downloaded from /aminet/dev/gui/gadtoolsbox20c.lha - Version 2.0c is needed.

This is version 1.62 of the extension and it's completely FREEWARE. If you really like this extension, send a post card to the author!

1.3 Installation

Installation

IMPORTANT! If you are upgrading from v1.5ß of the GUI extension, you must save all your programs as ASCII (see your AMOSPro manual) and reload them after installation as ASCII. I recommend you rename your old AMOSPro_GUI.Lib to GUI_v1.5 or something, so you can always get to it if you find a program you forgot to convert.

To install the GUI extension, follow these steps:

1) Copy the AMOSPro_GUI.Lib into your APSystem drawer.

2) Load AMOSPro and select "Set Interpreter" from the Config menu.

3) Select "Loaded Extensions".

4) Find and select extension slot 24.

5) Enter "AMOSPro_GUI.Lib" and select "Okay".

6) Save the configuration and quit AMOSPro.

Next time you run AMOSPro, the extension will be loaded. Sorry, but it's not compatible with the original AMOS.

You may also want to install the various GUI extension tools into the User menu. You can do this by following these steps:

1) Load AMOSPro.

2) Select "Add Option" from the bottom of the "User" menu.

3) At the requester, type in "GUI Convert" and select "Okay".

4) Select your newly created option from the "User" menu.

5) From the file requester, find and select "GUI_Conv.AMOS".

6) Select "Okay" from the next requester.

7) Press a key to cancel the requester.

8) Repeat steps 2 to 7 for the Hot Links editor program and the Image editor.

9) Now select "Save Configuration" from the Config menu.

1.4 Creating a GUI bank

Creating a GUI bank

Your GUIs are stored in AMOS banks and used in simple, easy-to-use commands. But first, to get to this stage, follow these steps:

1) Create your GUI in GadToolsBox using topaz/8 and save (NOT powerpacked!).

2) Select "Make GUI Bank" from the "User" menu (see Installation).

3) Locate your GUI file and load it.

4) Input a file name to save the GUI bank as.

Done! The GUI bank will also be copied into bank 20 of the current program.

The gadtools.library is a powerful library, but it has some missing features - Image buttons and Progress bars. These features have been added by the GUI extension:

Image Buttons

To create an image button, create a BUTTON gadget, but WITHOUT TEXT. When the GUI Converter finds this, it will ask you to load an Image Bank (created in the Images Grabber) containing the gadgets image.

When programming with this gadget, it acts exactly like a standard button gadget.

Progress Bars

To create a progress bar, create a TEXT gadget with the "Default" string set to "PBAR". The GUI Converter will do the rest!

When using this gadget in your programs, you can set the bar level in the range 0 to 100.

1.5 Using the Images Grabber

Using the Images Grabber

The Images Grabber utility allows you to grab the images you need for the custom gadgets of your GUI windows from an IFF picture. The picture is copied on a 640x256, Hires, 4 colour screen, so it could cut some parts or give things the wrong dimension. It's very easy to use: You just have to set the images with simple clicks of the mouse.

The program opens a little screen with 6 buttons inside:

2 arrows (up and down): You use these to decide which image must be cut or to see again the images already cut.

Scissors: This allows to grab the image with the number shown between the arrows.

Save coords: This saves the coordinates of the current images to the disk; You'll be able to load again them in the future to change them.

Load coords: This loads the coordinates of the images already saved on the disk. All the current images will be lost, but the picture will remain the same.

Arrow to a disk (Save): This saves the cut images. The program saves an Image bank (with a ".Img" extension) that will be asked for by the GUI Converter when you convert your GUI into an AMOS bank.

Restore: This deletes every image and the current picture, restarting the program from the beginning. Be careful using it because you could loose all your work.

Quit: This ends the program. Be careful using it because the program will end without asking you any confirmation.

The screen with the buttons can be moved clicking wherever you want and then moving the mouse up or down.

At the beginning a requester asks for an IFF picture where the images must be grabbed from. The picture is copied on a 640x256 - Hires - 4 colours screen (see above). The palette of the picture is changed automatically by the program in the palette of a normal Workbench screen.

These are some rules to using the program:

1) Cut 2 images for every custom gadget in this order: First the image of the not clicked gadget, then the one of the clicked gadget. Remember to draw the border of the gadget (if you want it....) because otherwise it won't be shown. The program automatically makes the second image as big as the first, but if you change the first one after having grabbed the second, this won't be changed.

2) Cut the images in the order of your custom gadgets because the GUI extension takes the first 2 images for the first gadget, then the second 2 for the second and so on. So remember the order of the gadgets!

3) If you click on the scissors to grab an image and then you don't want to do it any more, click the right button of the mouse.

1.6 Using the Hot Links Editor

Hot Links Editor

The Hot Links Editor helps you to make your GUI handling code much shorter, by adding keyboard shortcuts (Hot Keys) and linking gadgets together.

HotKeys

You can assign to each gadget any key using the Hot Links Editor. When the user presses the selected key, the associated gadget will be activated automatically and Gui Wait/Event will return the gadget number as if it were selected by the mouse. In this way, you don't have to check for pressed keys because the extension does it for you.

Links

This powerful ability was added when thinking about the most famous GUI frames for the Amiga, such as Magic User Interface and Class Act. With the link ability, you can make groups of up to 32 gadgets and then they will cooperate, working together. The link system works in this way:

You first choose the main gadget of the group (ROOT Gadget), then the other gadgets linked with that (LINK Gadgets).

When the ROOT gadget is activated by the user, the extension updates all the LINK gadgets with its new value automatically!

Here's a quick example:

There are two gadgets - a slider and an integer. The slider is the ROOT gadget and the integer is the LINK one. When the slider is moved, the integer will show its position in real time!

The advantage? You don't have to check anything in your Amos code, making it much neater! It works totally automatically! No longer do you don't need code like:

If Slider Moved Then Update Integer Gadget,Slider Position

This is a very simple example... you can create much more complex links! See GuiDemo.Amos for an example.

The editor

The Hot Links Editor is the tool used to create these links in a simple and fast way for your own GUIs. To use:

First of all you must load a GUI bank (If you run this editor as an accessory, the GUI bank of the current program is loaded automatically). Then, choose the GUI you wish to edit.

As soon as you select the GUI, the editor is split in to three Listviews:

ROOT GADGETS - Shows the list of ROOT gadgets.

LINK GADGETS - Shows the list of LINK gadgets associated to the current ROOT gadget.

UNSELECTABLE - Shows the list of gadgets that cannot be selected by mouse (Read below).

To select a ROOT gadget, simply click on it in your GUI with the mouse pointer. It will then be added to the list in the ROOT Gadgets Listview. Then choose the LINK gadgets, again by clicking on them with the mouse pointer.

When a gadget is selected, its type with its number are displayed in the Listview.

If you need to add a gadget that cannot be selected (for example a Text gadget or an empty Listview), you can do it choosing it in the UNSELECTABLE Listview.

When you've finished selecting LINK gadgets, you must click the close gadget of your GUI to end editing the hot links of that GUI. Now you can create/modify another GUI or save the whole new bank.

If you want to give a Hot Key to one of your ROOT gadgets, you just have to select that ROOT gadget and then click the "Hot Keys" gadget to give it a key combination.

The "Link On" gadget is used to decide what kind of link you want to set between the ROOT and its LINK gadgets. If the gadget is activated (the default state), the LINK gadgets will be updated according to the value/position of the ROOT gadget. If it's deactivated, the LINK gadgets will be ghosted or not ghosted according to the ROOT gadget value. If the ROOT gadget is other than 0, the LINKs will be activated, otherwise they'll be ghosted (see the checkbox gadgets in GuiDemo.Amos).

You can recognize "normal" links from the other ones because they are listed in the Listview in upper case, while the others in lower case.

When you finish your work with the Hot Links Editor, click on "Save" and the program generates a Link Bank (.Lnk) which is automatically loaded by the GUI Converter. This means that you have to remake your GUI bank once again with the GUI Converter!

1.7 Using the GUI Compiler

Using the GUI Compiler

Before the GUI Compiler can be used, run the GUI Patch program first. This creates a patched version of APCmp which enables a much more OS friendly AMOS.

Two versions of the GUI Compiler Shell are included: One for AMCAF users and one for non-AMCAF users.

The GUI Compiler shell is so easy to use and so similar to the original compilers, I don't think you need any instructions. :-)

1.8 Commands

Commands

Control

Gui Open

=Gui Close()

Gui Reset

Gui Bank

Gui Amiga

Gui Amos

Gui On/Off

Gui Lock/Unlock

=Gui Wait

=Gui Event

=Gui Window

=Gui Code

=Gui Code$

=Gui Read()

=Gui Read$()

=Gui Menu()

Gui Set

Gui Range

Gui Activate

=Gui Kind()

=Gui X/Y Gad()

=Gui Gad Width/Height()

Gui Title

Gui Move

Gui Resize

=Gui Width/Height()

=Gui In Width/Height()

=Gui Border()

=Gui Mouse X/Y

=Gui X/Y

=Gui Exist()

Gui Sensitive On/Off

=Gui Sx/y()

=Gui Sw/h()

Gui Remember On/Off

Gui To Front/Back

Gui Wait Vbl

Gui Beep

Gui Iconify/Uniconify

=Gui Gad Adr()

=Gui Os

Graphics

Gui Pen/Paper

Gui Writing

Gui Text

=Gui Len()

=Gui X/Y Font

Gui Ink

Gui Draw

Gui Bar

Gui Ellipse

Gui Cls

Gui Clw

Gui Gfx

=Gui Actual

Gui Scroll

Gui Paste Bob

Gui Paste Block

Gui Paste Icon

Requesters

=Gui Req()

=Gui Asl$()

=Gui Dir$/File$

=Gui Asl Screen

=Gui Asl Font

1.9 Gui Open

Gui Open

Instruction: Open a window.

GUI OPEN window,gui

GUI OPEN window,gui,bank

GUI OPEN window,gui,bank,x,y

GUI OPEN opens a window (number window) using the specified GUI data. You may also include a bank number from which to get the GUI data. This bank will then become the current GUI bank (see Gui Bank).

There is a third syntax where you can specify the position of the window. Otherwise it will default to the position as specified in the GadToolsBox editor.

As long as there's enough free memory, you can open as many windows as you wish.

When you open a window, this window becomes the current, selected window. If the window you specify is already open, it will be selected - no error will occur.

To close the window, use the Gui Close command.

An error will be reported if no bank has been specified and Gui Bank has not been called.

1.10 =Gui Close()

=Gui Close()

Function: Close a window.

=GUI CLOSE(window)

GUI CLOSE will close the specified window and returns a code which means...

0 - Window closed.

1 - First opened window closed.

2 - Last opened window closed.

3 - Last window closed.

Note: Windows will not close automatically when you select the close gadget on them. See the Gui Wait command to see how you can check for this.

Also see Gui Reset .

1.11 Gui Reset

Gui Reset

Instruction: Close ALL windows.

GUI RESET

GUI RESET simply closes all previously opened windows. It allows you to quickly end programs, or to clean everything up when called from Direct mode if your program crashes.

1.12 Gui Bank

Gui Bank

Instruction: Set the bank to be used by Gui Open.

GUI BANK bank

GUI BANK sets the bank that GUI data is stored, for future calls to Gui Open .

1.13 Gui Amiga

Gui Amiga

Instruction: Hide AMOS.

GUI AMIGA 0

GUI AMIGA 1

GUI AMIGA 0 simply carries out the AMOS instructions Amos To Back and Amos Lock. To get back, use Gui Amos .

GUI AMIGA 1 is more complex. It carries out the same commands as GUI AMIGA 0, but also does the AMOSPro compiler instruction Comp Test Off and Break Off. Also, it removes the AMOS interrupt routine. This makes your program more OS friendly, speeds up multitasking (AMOS progs generally slow it down) and stops a lot of the AMOS OS hacking that goes on.

However, under this mode, many AMOS routines (the ones that use interrupts) won't work. This includes all commands to do with AMAL, Music, Samples and Wait Vbl. If you need music and samples use another extension (AMCAF would be best!). Use Gui Wait Vbl instead of Wait Vbl.

With the GUI Compiler, this command makes AMOS code much more OS friendly and TOTALLY hides the fact that the program was written with AMOS!

Note that this only takes effect when your program is compiled and it must not be used with the Comp Test Off command!

1.14 Gui Amos

Gui Amos

Instruction: Reverse Gui Amiga.

GUI AMOS

GUI AMOS reverses what Gui Amiga does. It's the same as the AMOS instructions Amos Unlock and Amos To Front.

1.15 Gui On/Off

Gui On/Off

Instruction: Enable/Disable specified window.

GUI ON window

GUI OFF window

GUI ON and GUI OFF enable and disable the specified window. When a window is disabled, the user can't interact with it (i.e. can't select gadgets, etc.) and the busy pointer will be displayed (only with OS 3.x). GUI ON will turn the window back on again.

Also see Gui Lock/Unlock .

1.16 Gui Lock/Unlock

Gui Lock/Unlock

Instruction: Disable/Enable all windows except the specified one.

GUI LOCK window

GUI UNLOCK

GUI LOCK disables all the windows except the specified window. This is useful if you want to force the user to select a gadget in a certain window (e.g. a registration code, before you are allowed to use the program).

GUI UNLOCK enables all the windows.

To disable single windows, use Gui On/Off .

1.17 =Gui Wait

=Gui Wait

Function: Wait for the user to do something.

=GUI WAIT

GUI WAIT will wait until the user interacts with your program - your program will be frozen until something happens (you can stop it with Ctrl-C). The value it returns is more than or equal to 0, it is the number of the gadget selected. Otherwise, it's one of the following...

-1 - Close gadget selected.

-2 - Menu item selected.

-3 - Not used.

-4 - Raw key pressed.

-5 - Key pressed (without a text gadget selected).

-6 - You must uniconify the window!

-7 - Nothing selected (see Gui Event).

-8 - Window resized.

-9 - TCP/IP message (not yet implemented).

-10 - ARexx message.

If you have more than one window open, call Gui Window to find out which window (if any) this code applies to.

If the gadget selected is one that requires user input (e.g. text zone or listview), then use the commands Gui Code and Gui Code$ to get the inputted data.

With the ARexx message, you no longer need to use the AMOS command Arexx Wait. To catch the ARexx message, do something like:

```
G=Gui Wait
If G=-10
ARX=Arexx
Repeat
' Read the Arexx Message
MSG$=Arexx$(1)
[Parsing Arexx command]
' Answer back (see AMOSPro manual for details)
If ARX=1
Arexx Answer 0
Else
Arexx Answer$ 0,RETURN$
End If
ARX=Arexx
Until ARX=0
End If
```

Also see Gui Event .

1.18 =Gui Event

=Gui Event

Function: Return the current program status.

=GUI EVENT

GUI EVENT is similar to the Gui Wait command except it won't wait until something happens. It returns the same values as with Gui Wait but if nothing is selected when it's called, then -7 will be returned.

It's useful to check for user interruptions when a program is doing something, e.g. if the user presses STOP whilst downloading a file or rendering a picture. However, don't use Gui Event, when you could be using Gui Wait. e.g. don't use the following:

Repeat

SEL=Gui Event

If SEL<>-7 (Nothing selected)

' etc.

End If

Until SEL=-1

1.19 =Gui Window

=Gui Window

Function: Return the currently selected window.

=GUI WINDOW

GUI WINDOW returns the currently selected window. Mostly used with the Gui Wait command to create internally multitasking programs, i.e. all the windows seem to work independent from the others. This can be achieved by a program such as this:

Gui Amiga

Gui Open 1,1 : Gui Open 2,2 : Gui Open 3,3

Repeat

SEL=Gui Wait : WIN=Gui Window : Rem Get gadget and window

If WIN=1

_SELECTED_WINDOW1[SEL]

Else If WIN=2

_SELECTED_WINDOW2[SEL]

Else If WIN=3

_SELECTED_WINDOW3[SEL]

End If

Until SEL=-1 and WIN=1

Gui Reset : Gui Amos

Edit

,

Procedure _SELECTED_WINDOW1[SEL]

If SEL=-1

' Close gadget selected, so close window

NULL=Gui Close(1)

Else If SEL=0

' Selected gadget 0 in window 1

Else If SEL=1

' Selected gadget 1 in window 2

Else If SEL=2

' etc.

End If

End Proc

' etc.

With this method, you can create a separate procedure to handle input from each window your program has open.

1.20 =Gui Code

=Gui Code

Function: Return the status of the selected gadget.

=GUI CODE

GUI CODE is used to return the value entered/status of the last gadget selected (returned by Gui Wait). Depending on the type of gadget, the values returned are:

BUTTON - No value returned.

CHECKBOX - Boolean status (on or off).

INTEGER - Value typed in.

LISTVIEW - Selected item (NOT the data at that position).

MX - Selected item.

CYCLE - Currently selected item.

PALETTE - Number of the colour.

SCROLLER - Position.

SLIDER - Position.

STRING - None (see Gui Code$).

After GUI CODE is called, it is automatically reset to -1 until the next call to Gui Wait . If you need to read the value after this, use Gui Read .

1.21 =Gui Code$

=Gui Code$

Function: Return text from user input.

=GUI CODE$

GUI CODE$ returns the string entered/selected by user. This depends on the gadget which can be:

LISTVIEW - Selected data at position selected.

CYCLE - Selected item string.

STRING - Text entered.

Also see Gui Read$

1.22 =Gui Read()

=Gui Read()

Function: Return the status of the specified gadget.

=GUI READ(window,gadget)

GUI READ is similar to Gui Code except it doesn't need to be called after the specified gadget is selected. It will return the current status of the specified gadget from the specified window.

1.23 =Gui Read$()

=Gui Read$()

Function: Return text held in the specified STRING/LISTVIEW gadget.

=GUI READ$(window,gadget)

GUI READ$ is similar to Gui Code$ except it doesn't need to be called after the specified input.

1.24 =Gui Menu()

=Gui Menu()

Function: Return the selected menu item.

=GUI MENU(type)

GUI MENU will return the selected menu item, called after Gui Wait . The parameter type can be one of the following:

1 - Menu number.

2 - Item in menu.

3 - Sub-item from menu.

4 - Equals TRUE if selected other items.

In the case of other selected items, simply use the GUI MENU commands again to read them, as in the following example:

```
Repeat
SEL=Gui Wait
If SEL=-2 : Rem Selected menu
Repeat
Print "Menu"+Str$(Gui Menu(1)),"Item"+Str$(Gui Menu(2))
Until Gui Menu(4)=False : Rem Until no more selections
End If
Until SEL=-1 : Rem Until close gadget
```

1.25 Gui Set

Gui Set

Instruction: Set a gadget.

GUI SET window,gadget,attribute,value

GUI SET window,gadget,attribute,VARPTR(value$)

GUI SET window,gadget,attribute,ARRAY(array$(0))

GUI SET enables you to set a gadget on the specified window to a new value. It also forces gadtools.library to update that particular gadget. The attribute is one of the following for each gadget:

CHECKBOX - 0 to set boolean status (1 or 0, for on or off).

INTEGER - 0 to set value.

LISTVIEW - 0 to specify selected item.

1 to define an array to use for it (using ARRAY()).

2 to set the item at the top of the display.

MX - 0 to set selected item.

NUMBER - 0 to set number.

CYCLE - 0 to specify selected item.

PALETTE - 0 to set selected colour.

SCROLLER - 0 to set position of the bar.

1 to set total size.

2 to set visible size.

SLIDER - 0 to set position of the bar.

1 to define minimum value.

2 to define maximum value.

STRING - 0 to set string (using VARPTR()).

TEXT - 0 to set string (using VARPTR()).

PBAR - 0 to set progress level (0 to 100).

Note: For gadgets that require a string input (i.e. STRING and TEXT) the AMOS command VARPTR() has to be used, e.g. Varptr(NAME$). For setting the array to be used for the list, the AMOS command ARRAY() has to be used (single dimensional array only). e.g. Array(SURNAMES$(0)). Any empty elements (i.e. "") will be ignored and will not appear in the list. This array MUST be global - otherwise there'll be a GURU when the procedure ends (or you can close the window before ending the procedure).

If you wish to turn on/off ghosting of a gadget, simply enter attribute as -1, with either 1 or 0 as the new level. For example:

Gui Set 1,5,-1,1 : Rem Gadget is OFF

Gui Set 1,5,-1,0 : Rem Gadget is ON

1.26 Gui Range

Gui Range

Instruction: Limit input for integer gadgets.

GUI RANGE window,gadget,minvalue,maxvalue

GUI RANGE will limit input in integer gadgets to those specified.

For example, if you have done GUI RANGE 1,1,10,20, and the user inputs 5, it will automatically be set to 10. Similarly, if the user inputs 225373226, it will be set to 20.

1.27 Gui Activate

Gui Activate

Instruction: Activate an input gadget.

GUI ACTIVATE window,gadget

GUI ACTIVATE will activate the specified input gadget (string/integer), thus prompting the user to type something in.

1.28 =Gui Kind()

=Gui Kind()

Function: Return gadget type.

=GUI KIND(window,gadget)

Returns the kind of gadget as an index number, which is one of the following:

0 - BUTTON (with image)

1 - BUTTON

2 - CHECKBOX

3 - INTEGER

4 - LISTVIEW

5 - MX

6 - NUMBER

7 - CYCLE

8 - PALETTE

9 - SCROLLER

10 - reserved

11 - SLIDER

12 - STRING

13 - TEXT

14 - reserved

15 - PROGRESS BAR

1.29 =Gui X/Y Gad()

=Gui X/Y Gad()

Function: Return gadget position.

=GUI X GAD(window,gadget)

=GUI Y GAD(window,gadget)

These functions return the position of the specified gadget (in pixels), relative to the top left of the window.

Also see Gui Gad Width/Height which return the size.

1.30 =Gui Gad Width/Height()

=Gui Gad Width/Height()

Function: Return gadget size.

=GUI GAD WIDTH(window,gadget)

=GUI GAD HEIGHT(window,gadget)

These functions return the size of the specified gadget (in pixels).

1.31 Gui Title

Gui Title

Instruction: Set the title of the specified window.

GUI TITLE window,window_title$,screen_title$

GUI TITLE sets the title (window_title$) for the specified window as well as the name to be printed in the title bar when selected (screen_title$).

1.32 Gui Move

Gui Move

Instruction: Move a window.

GUI MOVE window,x,y

GUI MOVE will move the specified window to the new screen coordinates.

1.33 Gui Resize

Gui Resize

Instruction: Resize a window.

GUI RESIZE window,width,height

GUI RESIZE resizes the specified window to the new size (in pixels).

1.34 =Gui Width/Height()

=Gui Width/Height()

Function: Return size of window.

=GUI WIDTH(window)

=GUI HEIGHT(window)

GUI WIDTH and GUI HEIGHT return the width and height of the specified window.

1.35 =Gui In Width/Height()

=Gui In Width/Height()

Function: Return the internal size of window.

=GUI IN WIDTH(window)

=GUI IN HEIGHT(window)

GUI IN WIDTH and GUI IN HEIGHT return the width and height of the specified window, excluding the window borders (returned from Gui Border).

1.36 =Gui Border()

=Gui Border()

Function: Calculate the size of a window's border.

=GUI BORDER(window,border)

GUI BORDER returns the size of the border of each of the four sizes of the specified window. The border parameter can be:

0 - Left border.

1 - Top border.

2 - Right border.

3 - Bottom border.

1.37 =Gui Mouse X/Y

=Gui Mouse X/Y

Function: Get the coordinates of the mouse.

=GUI MOUSE X

=GUI MOUSE Y

GUI MOUSE X and GUI MOUSE Y will return the screen coordinates of the mouse. If you want them relative to a window, you'll have to use Gui X/Y .

1.38 =Gui X/Y

=Gui X/Y

Function: Return the coordinates of a window.

=GUI X

=GUI Y

GUI X and GUI Y return the screen coordinates of the specified window.

1.39 =Gui Exist()

=Gui Exist()

Function: Check if a window is open.

=GUI EXIST(window)

GUI EXIST checks if the specified window is opened. If it isn't, then FALSE is returned. Otherwise, the window's structure address will be reported. DON'T fiddle with this structure unless you really know what you're doing!

1.40 Gui Sensitive On/Off

Gui Sensitive On/Off

Instruction: Turn on or off font sensitivity.

GUI SENSITIVE ON

GUI SENSITIVE OFF

GUI SENSITIVE ON will make it so your programs use the WB font, adapting the window and gadgets sizes to the new font size. If the resize causes the window to no longer fit on the screen, this command will have no effect. This is the default setting.

GUI SENSITIVE OFF makes it so the topaz/8 font is used. When you create your GUI with GadToolsBox, you MUST use the topaz/8 font.

Note: Custom gadgets are NOT font sensitive.

1.41 =Gui Sx/y()

=Gui Sx/y()

Function: Return the position of a point, with font sensitivity.

=GUI SX(x)

=GUI SY(y)

GUI SX and GUI SY will return the new position of a point, when scaled as the gadgets are with Gui Sensitive On .

For example, if you were to do Gui Bar 10,15 To 25,30 with the topaz/8 font, you should do Gui Bar Gui Sx(10),Gui Sy(15) To Gui Sx(25),Gui Sy(30) instead, if you put font sensitivity on (it is by default).

1.42 =Gui Sw/h()

=Gui Sw/h()

Function: Return width and height, with font sensitivity.

=GUI SW(width)

=GUI SH(height)

GUI SW and GUI SH return the width and height, rescaled by the font sensitivity routine. See Gui Sx/y for more details.

1.43 Gui Remember On/Off

Gui Remember

Instruction: Turn on or off remembering window positions.

GUI REMEMBER ON

GUI REMEMBER OFF

GUI REMEMBER ON will make it so when a window is closed, it's position is remembered. When and if that window is reopened, the window is opened at the position that it was closed.

GUI REMEMBER OFF sets it back to the default setting, where windows are always opened at the position set in the GadToolsBox editor.

1.44 Gui To Front/Back

Gui To Front/Back

Instruction: Move window to front or back.

GUI TO FRONT window

GUI TO BACK window

GUI TO FRONT moves the specified window to the front of the display. GUI TO BACK moves it to the back.

1.45 Gui Wait Vbl

Gui Wait Vbl

Instruction: Wait for the next vertical blank period.

GUI WAIT VBL

GUI WAIT VBL vbls

GUI WAIT VBL is exactly like the AMOS Wait Vbl command, except for intuition. A parameter can be added which causes the program to wait for the specified vbl count.

1.46 Gui Beep

Gui Beep

Instruction: Flash the screen.

GUI BEEP

GUI BEEP flashes your screen (and plays a beep or sample, depending on your Workbench preferences) to alert the user.

1.47 Gui Iconify/Uniconify

Gui Iconify/Uniconify

Instruction: Iconify/Uniconify a window.

GUI ICONIFY window

GUI UNICONIFY window

GUI ICONIFY iconifies the specified window. GUI UNICONIFY uniconifies it, back to it's original state. Simple eh?

1.48 =Gui Gad Adr()

=Gui Gad Adr()

Function: Get the structure address of a gadget.

=GUI GAD ADR(window,gadget)

GUI GAD ADR returns the structure address of the specified gadget. Use it at your peril.

1.49 =Gui Os

=Gui Os

Function: Return the operating system version.

=GUI OS

GUI OS simply returns the operating system version.

1.50 Gui Pen/Paper

Gui Pen/Paper

Instruction: Set the pen and paper colours.

GUI PEN colour

GUI PAPER colour

GUI PEN and GUI PAPER sets the colour to be used for future Gui Text calls. They act just like the AMOS Pen and Paper commands.

1.51 Gui Writing

Gui Writing

Instruction: Change graphics writing mode.

GUI WRITING bitpattern

GUI WRITING sets the graphics drawing mode, just as the AMOS command Gr Writing does. The bitpattern can be made up with the following:

%001 - Replace any existing graphics with new graphics.

%010 - Change old graphics that overlap with new graphics using XOR.

%100 - Reverse ink and paper colours, creating inverse video effect.

That was just copied out of the AMOSPro manual - for further details, I suggest you look at that! :-)

1.52 Gui Text

Gui Text

Instruction: Print text in window.

GUI TEXT x,y,string$

GUI TEXT prints string$ at x,y on the current graphics output.

1.53 =Gui Len()

=Gui Len()

Function: Get text length in pixels.

=GUI LEN(text$)

GUI LEN simply returns the length of the text in pixels - just like the AMOS function, Text Length().

1.54 =Gui X/Y Font

=Gui X/Y Font

Function: Get font size.

=GUI X FONT

=GUI Y FONT

These functions return the size of the font.

1.55 Gui Ink

Gui Ink

Instruction: Set drawing colour.

GUI INK colour

GUI INK sets the colour to be used to subsequent drawing operations.

1.56 Gui Draw

Gui Draw

Instruction: Draw a line.

GUI DRAW x,y TO x1,y1

GUI DRAW TO x1,y1

GUI DRAW draws a line from x,y to x1,y1. It's just like the AMOS version.

1.57 Gui Bar

Gui Bar

Instruction: Draw a filled bar.

GUI BAR x,y TO x1,y1

GUI BAR draws a filled bar from x,y to x1,y1 - just like the AMOS equivalent.

1.58 Gui Ellipse

Gui Ellipse

Instruction: Draw an ellipse.

GUI ELLIPSE x,y,width,height

GUI ELLIPSE draws an ellipse with centre point x,y and specified radices of width and height.

1.59 Gui Cls

Gui Cls

Instruction: Clear the current graphics output.

GUI CLS colour

GUI CLS clears all of the current graphics output (including window borders!) with the specified colour. It's really there for screen support which will come in a future version.

1.60 Gui Clw

Gui Clw

Instruction: Clear a window.

GUI CLW window,colour

GUI CLW clears the specified window of all all graphics.

1.61 Gui Gfx

Gui Gfx

Instruction: Set the following graphics output.

GUI GFX type,number

GUI GFX sets where following graphics operations are carried out. The parameter type, can be:

0 - Window.

1 - Screen (not yet implemented).

The parameter number tells which window or screen to set it to. For example, to set the current output to window 7...

Gui Gfx 0,7

1.62 =Gui Actual

=Gui Actual

Function: Return the current graphics output window.

=GUI ACTUAL

This function returns the window number, set by Gui Gfx .

1.63 Gui Scroll

Gui Scroll

Instruction: Scroll a graphical area.

GUI SCROLL x,y TO x1,y1,dx,dy

GUI SCROLL scrolls the area x,y to x1,y1 by dx pixels horizontally and dy pixels vertically.

1.64 Gui Paste Bob

Gui Paste Bob

Instruction: Paste an AMOS bob.

GUI PASTE BOB image,x,y

GUI PASTE BOB pastes a bob in the current graphics output. It acts just like AMOS's Paste Bob command.

1.65 Gui Paste Block

Gui Paste Block

Instruction: Paste an AMOS block.

GUI PASTE BLOCK block,x,y

GUI PASTE BLOCK pastes a block in the current graphics output.

1.66 Gui Paste Icon

Gui Paste Icon

Instruction: Paste an AMOS icon.

GUI PASTE ICON icon,x,y

GUI PASTE ICON pastes an icon in the current graphics output.

1.67 =Gui Req()

=Gui Req()

Function: Open a requester.

=GUI REQ(title$,message$,gadget$)

GUI REQ creates a requester with little effort. It returns the gadget selected by the user.

To have multiple gadgets, you must separate each one with a "|" character. Also, to have more than one line in the message, you have to put in Chr$(10)s. For example:

MSG$="Please insert a bun"+Chr$(10)+"in any drive"

GAD$="OK|How about a cake?|I've eaten the buns|No!"

SEL=Gui Req("Oh no!!",MSG$,GAD$)

If the right-most gadget is selected, 0 will be returned, otherwise 1 upwards will be returned starting from the left gadget. So, for the previous example, SEL would be:

1 - OK

2 - How about a cake?

3 - I've eaten the buns

0 - No!

1.68 =Gui Asl$()

=Gui Asl$()

Function: Call the standard ASL file requester.

=GUI ASL$(title$,dir$,file$,pattern$)

GUI ASL$ causes the ASL file requester to pop up for you to select a file from. The file name (with path name in front) is returned, or an empty string if cancel is selected.

You can get the path and file names on their own with GUI DIR$/FILE$.

1.69 =Gui Dir$/File$

=Gui Dir$/File$

Function: Get path and file name from Gui Asl$().

=GUI DIR$

=GUI FILE$

GUI DIR$ and GUI FILE$ return the path and file name, from the previous call to Gui Asl$() . If cancel was selected, these will return empty strings.

1.70 =Gui Asl Screen

=Gui Asl Screen

Function: Call the ASL screen mode requester.

=GUI ASL SCREEN

GUI ASL SCREEN opens the ASL screen mode requester and returns the DisplayID value of the selected mode.

1.71 =Gui Asl Font

=Gui Asl Font

Function: Call the ASL font requester.

=GUI ASL FONT

GUI ASL FONT opens the ASL font requester and returns the selected font.

1.72 FAQ

Frequently Asked Questions

Q. Can I open Intuition/AGA screens?

A. Actually no... these commands are under development together the Amos Screen Promoter project.

Q. After the installation of the Gui Extension the compiler doesn't work, or it reports some strange errors.

A. This isn't a bug of the GUI extension!!! The Compiler shell is bugged! It doesn't handle extensions with a number greater than 20. The error is very easy to fix. Follow these steps:

1) Load the Compiler_Shell.Amos program.

2) Open all the procedures.

3) Go to line number 573.

4) Replace "Until R<-26" with "Until R<-42".

5) Save the program.

6) It's done! :)

The same error is in the Tiny_Shell.Amos program at line 288.

Q. When using the GUI Compiler, my programs have a corrupted display... why?

A. The GUI Compiler removes the Amos patches and other hacks. This means that you CAN'T use Amos graphics... AMOS MUST BE ALWAYS BE AT THE BACK! You may only use Intuition. You should put the Gui Amiga 0 or 1 command at the start of your program. If you aren't interested in multitasking and OS compatibility use the standard AMOSPro Compiler.

1.73 History

History

v1.62 04-05-97

- Fixed a nasty bug in the Custom string handling routines. Special thanx to Steve Clack (Liquid Software Design) and John C. Bintz for the bug reports

- Updated the font sensitivity routines

v1.61 30-03-97 Official release

- New Commands:

+ Gui Os

+ Gui Sx

+ Gui Sy

+ Gui Sw

+ Gui Sh

+ Gui Clw

+ Gui X Gad

+ Gui Y Gad

+ Gui Gad Width

+ Gui Gad Height

+ Gui Kind

+ Gui Gad Adr

+ Gui Range

+ Gui X Font

+ Gui Y Font

+ Gui Len

+ Gui Activate

- Bug fixed that caused window to be trashed if iconified several times.

- Gui Wait Vbl command expanded to take a parameter.

- Added AREXX support!

- Fixed stupid bug in the Gui Converter that may cause Enforcer Hits.

- Now you can break the Gui Wait command :) (Only if there is a GUI opened).

- Fixed Gui Sensitive problem! Now the routine handle correctly small fonts such as xen/8.

- Now font sensitive is activated as default.

- Changed/added Gui Wait values:

-3 - Internal use! No longer returned!

-9 - TCP/IP message received (not yet implemented).

-10 - Arexx message received.

- New Gui Compiler/Shell v1.22.

- AMOS string handling routines have been altered so all strings are automatically NULL terminated!

- Fixed bug: You can now set/range an integer gadget to a negative value.

v1.6 24-08-96

- New commands:

+ Gui Key Shift

+ Gui Read

+ Gui Read$

+ Gui Remember On/Off

+ Gui Beep

+ Gui On/Off

+ Gui Lock

+ Gui Unlocks

+ Gui Title

+ Gui In Width

+ Gui In Height

+ Gui Paper

+ Gui Pen

+ Gui Writing

+ Gui Text

+ Gui Event

- Fixed the 2 Enforcer hits caused by listview gadgets when a window is closed.

- Fixed another bug in the Gui Close command... now ALL the allocated memory will be released.

- Fixed bugs in the listview gadgets management routine. Now the ShowSelected Options works fine, and no more random elements appear in the listview.

- Fixed a bug in the Gui Converter... now the GUI banks are a bit shorter.

- Fixed/updated other minor aspects of the GUI Converter.

- Now Gui Wait doesn't lock the system if AMOS is to front, or if you press Amiga-A just before the Gui Wait command. e.g. If AMOS is at front, Gui Wait puts it to back!

- Fixed bug in the progress bar gadgets. Now the gadget ALWAYS works fine.

- Fixed bug in the font sensitive routine.
- Now the custom gadgets are better located in a font sensitive GUI.
- NEW!!! added the HotLinks! ... real power. :)
- Now Gui Asl$ allows pattern matching.
- Improved the Gui Amiga command.
- Improved the Gui Open command:
+ Now you can set the x,y coordinates of a GUI before opening it.
+ If you try to open a already open window, this window will be put to front.
+ Gui Open checks if the GUI bank is supported or if it’s a old version.
- Now Gui To Front activate the window too.
- Now Gui Wait report windows resizing.
- Added a new version of the GUI Image Grabber (2.1) (Thanx Michele!).
- Added a new version of the GUI Compiler (1.2).
+ More OS integration.
+ More stable under OS 3.x.
+ NEW: hides the AMOS nature of the program! Now it’s impossible to
discover AMOS with Artm/SnoopDos/Scout/Xoper etc.!
+ NEW: purify option! Now the executable is "purified" from AMOS related
things... if you Zap the code you will find nothing!
+ Removed silly (bugged) amos patch/hack.
- Added a new version of the GUI Converter (1.6).
- No longer sets the current directory to RAM: at boot.
v1.5ß First Public BETA VERSION

1.74 Credits

Credits
Programming:
Pietro Ghizzoni
E-Mail:
ghizzo@agonet.it
Or leave a message at:
Amy Professional BBS (ITALY AMOS USER CLUB) 049-604488,
Or Snail-Mail:
Pietro Ghizzoni
Via Termine 70/b
29017 Fiorenzuola D’Arda
Piacenza - Italy
Any problems or ideas? He’s the person to contact!

Click here for his only piece of text in this guide

Manual:

Ben Wyatt

E-Mail:

bwyatt@paston.co.uk

I don't think you need my address or anything. :-)

If you have found any mistakes in this doc, let me know!

I'll just take this opportunity to advertise two of my games:

game/2play/KnockOut2.lha - Addictive 0-8 player game! v3 coming soon!

game/demo/BorisBall.lha - Fast'n'frantic bat'n'ball game.

1.75 Future

These features are planned for the future:

- AGA-Screen support.

- AMOS Screen Promotion.

- More gfx commands.

- XFA (eXtra Fast Animation) support. Yeaahhh! :-)

- Advanced (easy to use) set of copper handling commands.

- A new set of powerful commands for internet connection.

- A new set of fast asynchronous(!!) file handling commands.

- A new set of commands for TAG's handling.

- The GUI Manager accessory.

- Windows resize.

- ... Any good ideas you may have. ;)

1.76 Credits and greetings

Special credits to:

- Ben Wyatt: Thanx for your great support, patience, ideas and collaboration!

- Michele Berionne: You are a REAL friend! :) Thanx for your unreplaceable support!

- Giovanni Petrella (Flynx): How are you? I'm waiting new AMOS programs from you..... very good as usual :) Amiga RULEZ!!

- Fiorenzo Deremigio (Firer): Hey! The FIRST Gui extension beta-tester! Thanx you!

- Michael Cox: Thanx for your great work!

- Mik of ClassX: Thanks for your useful programming tips and tricks!

Greetings to:

Fabio Chiechi, Lorenzo Succi, Luca Ferraris, Chris Hodges, Chris Wostelnholme, Braneloc, Paul Hickman, Fabrizio Bazzo, Doug Kanora, Andy Kellet, Nikola Smolensky, Palle(Balls) Larsen :), Pawel Rutkowski, Scott Wade, Philip Noworita, Petri Hakkinen, Bernardo Innocenti, Flavio Stanchina, Sergio Ruocco, Ferruccio Zamuner, Steve Clack, John C. Bintz.

!!!

o o

+----------oOO-(_)-OOo----------+

| |

| Pietro Ghizzoni - Dairymen Soft __ /// Amiga 12OO |

| E-Mail: ghizzo@agonet.it \\/// ’O3O 5Ohz |

| Team AMIGA \/// 1OMB - CD4x |

| |

| Amos Professional Team Coordinator AMIGA RULEZ!! |

| |

+----------+