StormWIZARD

COLLABORATORS			
	TITLE : StormWIZARD		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		March 28, 2025	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

StormWIZARD

1.1 StormW.guide

StormWIZARD 2.0

Contents

1.2 StormWIZARD.guide/ST_Order

Please print the enclosed form on your printer.

Check the desired products and fax or send us the completely

filled-out form.

Our address:

HAAGE & PARTNER COMPUTER Ltd.

PO Box 80

61191 Rosbach v.d.H.

Fax: +49 - 6007 - 75 43

Order(mark the desired item(s))

* Yes, send me the English version of StormC at a price of 598 DM

* Yes, please send me StormWIZARD at a price of 149 DM

If you do not live in Germany you have to pay in advance plus

30 DM for shipping.

First name: ______________________________________

Name: ______________________________________

Street: ______________________________________

Zip code: _________ Town: ______________________________________

Country: ____________________________

Telephone: ______________________________________

E-mail: ______________________________________

Per enclosed cash or advance-check.

1.3 StormWIZARD.guide/ST_CRIGHT

1.4 StormWIZARD.guide/ST_Lizenz

License agreement

1 In general

(1) Object of this contract is the use of computer programs from

the HAAGE & PARTNER COMPUTER Ltd., including the manual as well

as other pertinent, written material, subsequently summed up as

the product.

(2) The HAAGE & PARTNER COMPUTER Ltd. and/or the licensee indicated

in the product are owners of all rights of the products and the trademarks.

2 Right of usufruct

(1) The buyer does receive a non-transferable, non-exclusive right
to use the acquired product on a single computer.

(2) In addition the user may produce one copy for security only.

(3) The buyer is not legitimated to propagate the acquired product, to
rent, to offer sublicenses or in any other way to put these at the
disposal of other persons.

(4) It is forbidden to change the product, to modify or to
re-assemble it. This prohibition includes translating, changing,
re-engineering and re-use of parts.

3 Warranty

(1) The HAAGE & PARTNER COMPUTER Ltd. guarantees that up to the point
in time of the delivery the data carriers are physically free of
material and manufacturing defects and the product can be used as
described in the documentation.

(2) Defects of the delivered product are removed by the supplier within
a warranty period of six months from delivery. This happens through
free replacement or in the form of an update, at the discretion of
the supplier.

(3) The HAAGE & PARTNER COMPUTER Ltd. does not guarantee that the
product is suitable for the task anticipated by the customer.
The HAAGE & PARTNER COMPUTER Ltd. does not take any responsibility
for any damage that may be caused.

(4) The user is aware that under the present state of technology it is
not possible to manufacture faultless software.

4 Other

(1) In this contract all rights and responsibilities of the contracting
parties are regulated. Other agreements do not exist. Changes are
only accepted in written form and in reference to this contract and
have to be signed by both parties.

(2) The jurisdiction for all quarrels over this contract is the court
responsible at the seat of HAAGE & PARTNER COMPUTER Ltd.

(3) If any single clause of these conditions should be at odds with the
law or lose its lawfulness through a later circumstance, or should
a gap in these conditions appear, the unaffected terms will remain
in effect. In lieu of an ineffective term of the contract or for the
completion of the gap an appropriate agreement should be formulated
which best approximates within the bounds of the law the one that

the contracting parties had in mind as they agreed on this contract.

(4) Any violation of this licence agreement or of copyright and trademark

rights will be prosecuted under civil law.

(5) The installation of the software constitutes an agreement with these

license conditions.

(6) If you should not agree with this license agreement you have to

return the product to the supplier immediately.

September 1995

1.5 StormWIZARD.guide/ST_Welcome

Welcome to the new Amiga GUI editor.

This demo version of StormWIZARD shows you the ability of an advanced GUI editor and a Boopsi library.

There are many libraries that offer such functions. But some of them are not focused on the main problems and others are not as flexible. Besides many of them do not have a visual editor to guarantee ease of use.

Besides this all these libraries have one problem: they are Freeware, Public Domain or Shareware. That way you will never know if its development will continue in the future. And what about the PowerPC Amiga ? Are they written in Assembler or any other programming language that will not be ported to PPC ?

Using StormWIZARD you can be sure that there will be a PPC-Native version, because we are developing the PowerPC Development System for the Amiga that naturally uses Wizard.Library.

StormWIZARD is another part in our professional development system StormC. It can also be used as a stand-alone product in combination with other compilers or programming languages; it is the perfect supplement for StormC. StormWIZARD will integrate itself perfectly into the project manager.

In version 1.1 of StormC the project manager got a new section for ".wizard" files and the linker is now able to bind the ".wizard" resources to your program code as binaries. This option is only available with StormC.

The advantages of StormWIZARD are short turnaround times and a radical gain in programming efficiency.

1.6 StormWIZARD.guide/ST_Philo

Philosophy of StormWIZARD

A very large chunk of your programming time is usually eaten up by design and maintenance of the user interface.

Most of all, maintenance is a difficult and error-prone activity because there are complex structs that cause errors that are often hard to find.

When working with StormWIZARD you will never see the source of your GUI. There will be a resource file that contains all the information.

One big advantage of this method is its independence of the programming language. You are not forced to use ANSI-C, C++ or Assembler. The resources produced by StormWIZARD can be used with any programming language.

The second advantage and certainly the more important one is that you do not make changes in the source but with StormWIZARD in the resource file. The GUI code is strictly separated from your program source. There is no need to start the compiler to have a look at the changes. StormWIZARD will show you the latest appearance of your GUI at the touch of a button. This makes it much easier to design and change your program's GUI.

You can concentrate on programming while StormWIZARD takes care of your user interface.

1.7 StormWIZARD.guide/ST_Maschine

Requirements

You will need the following items to run StormWIZARD:

- any Amiga with hard disk
- Kickstart/Workbench 3.0 (v39) or better
- 2 MB RAM

1.8 StormWIZARD.guide/ST_Install

Installation

The installation of StormWIZARD is done with the Installer. Double-click the installation icon and follow the instructions.

If there are any questions, please contact:

HAAGE & PARTNER Computer GmbH

Mainzer Strasse 10A

61191 Rosbach

Germany

Phone: +49 6007 930050

Fax : +49 6007 7543

E-Mail: haage_partner@compuserve.com

WWW: http://ourworld.compuserve.com/homepages/haage_partner

1.9 StormWIZARD.guide/ST_Tutorial

Tutorial

The tutorial of the original manual will show you everything about working with StormWIZARD. You will learn how to use the editor, how to place gadgets and how to create a menu.

1.10 StormWIZARD.guide/ST_Start

Starting StormWIZARD

The following exercise will show you how to create windows, how to add gadgets and how to change their size and position.

Please start StormWIZARD by double-clicking it's icon.

After starting the program the window editor appears. It will show all parts of your project (windows, gadgets, menus) in a list.

Initially this list is empty because you have not created any window yet.

First Exercise

Create a new window by clicking on "Add Window".

Enter the object name of the window and confirm with OK. Now the first entry appears in the list of the window editor.

Clicking on "+" or <Space> of the first entry will show two more entries. A double click on them will lead you to the Gadget or the Menu editor.

These entries can also be selected by using the cursor keys <UP> und <Down> followed by <Return>.

Now double-click "Gadgets" to go to the Gadget editor.

Now click the gadget "Add Object" with the left mouse key and keep it pressed. A Popup menu will appear. Select the entry "Date".

Now the Date attributes requester appears; ignore for it now and click on "Ok" or press the <O> key.

Click on "Redraw" and you will get your first Wizard window. That may be impressive enough, but it is not all StormWIZARD can do for you.

Please add another object by selecting HGroup (horizontal group) from the popup menu "Add Object". Click on "Ok".

Clicking on "Redraw" will not show a different GUI, but when you are resizing the window you will see that the date gadget is always at the upper edge. This is caused by the higher priority of the group gadget which drives away the date gadget.

The arrangement of the dimensions of the group gadget is strongly dependant on its priority. Our invisible group gadget is the root gadget which is added to every window by the editor.

Please select (not double-click) the date gadget now. Click on {b}"Down". Now the date gadget is below the group gadget and it is part of this HGroup (horizontal group).

Now the root group has only one object - the HGroup gadget. This one has a "child", the date gadget.

Double-click the HGroup gadget in the gadget editor list. The attribute requester appears. Click on "Attribute".

Look for the Integer gadget "HBorder" and enter "10" as value. Confirm with <Ok>. Now click on "Redraw" again to have a look at the changes.

Minimise the window and have a look at the right and left borders of the date gadget. The are both 10 pixels. That is what we entered at "HBorder".

Now double-click the HGroup gadget in the Gadget editor list and select the entry "VBorder". You will find it in the upper right column. Enter "6" and confirm. Click on "Redraw" and watch the distance to the top and bottom. It will be 6 pixels - it’s magic ;-).

1.11 StormWIZARD.guide/ST_Project

Second Exercise

To get a border around the members of a group, double-click the HGroup gadget and select the border type at the attributes.

Select the item "SButton" from the popup menu and then activate "Redraw".

Drag the window to minimal size. Now there is a border around the DATE gadget, but it is drawn into the gadget. To avoid this you must set the attribute of the group gadget "BHOffset" to 5. Activate "Redraw" to see the changes.

Now the border is wider. It begins 5 pixels in front of the old position and ends 5 pixels after it. But still the border of the DATE gadget is not correct. To change this you should set the parameter "BVOffset" to 4. Activate "Redraw" and have a look at the result - now everything is fine.

1.12 StormWIZARD.guide/ST_Project

An Example with Paging and Notification

Select the menu item "New" at the window editor to set the default.

Select "Add Window" and enter a name for the window. Click on "Ok" to close the attribute requester.

Clicking on the first entry’s "+" or pressing <Space> while the first entry is active will show two more entries. A double click on either of them will lead you to the Gadget or the Menu editor.

These entries can also be selected by using the cursor keys <UP> und <Down> followed by <Return>.

Enter the GADGET editor and click on "Add Object". Select "HGroup" and {b}"Ok".

Create a new cycle object by clicking on "Add Object" and selecting the entry "Cycle". Switch to page "Labels" and create two new "Cycle" entries:

Page 1

Page 2

Press <OK>.

The new cycle gadget will be added to the horizontal group automatically.

Create a new HGroup gadget. A requester appears; select the "Attribute" page.

Enter 1 at "Max. Pages:". Press "Ok".

Press "Down" to move the new horizontal group to the root gadget. The HGroup gadget is no longer connected with the first HGroup object.

Insert a new BUTTON object. Enter "At Page 1" into the field "Name"_{ub}. Press "Ok".

Create another BUTTON object with the text "At Page 2" and set the value "Page:" to 1.

Now click "Redraw" to have a look at your work.

Using the cycle gadget in the Preview window has no effect, because there is no connection between the cycle gadget and the group gadget. Therefore you will have to create one.

Select the cycle gadget in the GADGET editor and click on "Notify Editor".

Please click on the popup at the upper right corner and select item "3, HGroup". A connection object is created.

This object has the correct settings as default. Now close the NOTIFY editor with "Ok" and press "Redraw".

There is no visible difference, but now the cycle gadget can be used for "Paging".

Let’s try some other connections to demonstrate the possibilities of Notification.

Select entry "3, HGroup" in the GADGET editor. Using the NOTIFY editor you should create a connection to the cycle gadget now. Press "Ok".

Double-click on item "3, HGroup" in the GADGET editor. Add two new labels with the text "Page 1" and "Page 2".

Select the "Attribute" page and change the value of "HBorder" to 10. Change "VBorder" to 6, "BHOffset" to 6 and "BVOffset" to 4.

Set "Max. Pages" to Zero to indicate that it is not used here anymore (labels define the number of pages now).

Press "Redraw" and have a look at the results.

1.13 StormWIZARD.guide/ST_Project

Layout rules

The layout processes described before can be summarized in 11 rules:

1. Rule:

A horizontal grouping gadget arranges its members next to each other while a vertical grouping gadget arranges them below of each other.

2. Rule:

Freiraeume werden immer zwischen den Mitgliedern dargestellt und je nach Gruppen-Gadgettyp auch berechnet. D.h. ein horizontales Gruppen-Gadget wird einen Freiraum zwischen den nebeneinander liegenden beiden Objekten erzeugen. Die Anzahl der Objekte um 1 verringert ergibt die Anzahl der Freirume. Bei nur einem Mitglied kann kein Freiraum erzeugt werden.

3. Rule:

The minimum width of a horizontal group as visible to its parent group is calculated by adding each objects minimum object widths and the empty space between these objects. Additionally, the HBorder value must be added twice. The minimum width results from the double VBorder value and the highest minimum height of an object.

4. Rule:

The minimum width of a vertical group as visible to its parent group is calculated from the double HBorder value and the highest minimum width of an object. In contrast, the minimum height of a group is calculated by adding each objects minimum height and the empty space between these objects. The VBorder value is once again added twice.

5. Rule:

Within a horizontal group a member always inherits the grouping gadgets full height. Of course this includes the VBorder value above and below of the gadget. A vertical grouping gadget assigns its members the equal size, respecting the HBorder value to the left and the right.

6. Rule:

A horizontal grouping gadget calculates a members width using its priority and its minimum width while respecting the minimum width of other members. Additional space is added to the left and the right of a member accordingly to its HBorder value.

Accordingly, vertical grouping gadgets add the VBorder value above and below of each member and use the remaining value to calculate its height. The minimum height and priority of each object is considered as well as the total height value of all members. Thus, the decisive attribute for the distribution of the available space among the members of a group is the priority between these objects!

7. Rule:

If a groups member has a priority of 0 this means that within an horizontal group its width will always be the smallest value possible! Within horizontal groups this applies to the height of a member.

8. Rule:

If a horizontal grouping gadget has the "Equal Size" flag set this means that all members are assigned the same minimum width. To achieve this, the grouping gadget asks all members for their minimum width and continues its calculations with the highest minimum width for all members. The vertical grouping gadget does the same for the minimum height.

9. Rule:

If the total priority of all members of a group is 0 the VarSpace attribute must be initialized with 0 or 100 percent only. If you nevertheless specify a different value you're relying on undefined positioning. In future versions of the library this may change !

10. Rule:

An object always attaches itself to its parent group. This occurs using the page specified in the members attributes. If an object has a page value of 1, it will be visible only if the first page is the grouping gadgetÕs current page.

11. Rule:

The Dock mode of a grouping gadget respects a members minimum width and height only: "Space" and "Varspace" are also not taken into account. For this mode the grouping gadget must be.

1.14 StormWIZARD.guide/ST_Project

Class Specific Layout Behaviour:

Each class uses the available space in a different manner. There are two main groups.

* Classes which use the available space completely:

- Group Classes
- Scroller Classes
- Slider Classes
- ListView Classes
- Arrow Classes
- Line Classes
- Colorfield Classes
- VectorButton Classes
- Space Classes
- Image Classes
- ImageButton Classes
- ImageToggle Classes
- ImagePopup Classes
- Palette Classes
- VectorPopup Classes
- Hierarchy Classes

* Classes with a fixed height which can be centred vertically:

- Button Classes
- String Classes
- Label Classes
- CheckBox Classes
- MutualExclusion Classes
- Integer Classes
- Toogle Classes
- Args Classes
- Gauge Classes
- Date Classes
- Cycle Classes
- TextPopup Classes

These classes must have a priority of Zero in a vertical group. For compatibility reasons this should not be changed.

1.15 StormWIZARD.guide/ST_Project

Objectcommunication

The Notify objects

You can for example combine a slider with an integer gadget. This requires the creation of a special link object. Thats the purpuse of the Notify Editor.

The link object can modify information identifiers in such a way that the integer gadget recognizes this information as a class attribute.

For example, the object allows you to convert a messages WSLIDERA_Top identifier, which contains the current position of a slider, into WINTEGERA_Long. This new identifier is understood by our destination object, the integer gadget. Basically you can connect all classes in such a way. However, if the user does not adjust the slider but instead changes the value of the integer gadget, the slider object is not notified. For this reason you should create another link object which converts the WINTEGER_Long identifier into WSLIDERA_Top. Hereby, both objects are linked with each other in both directions. Note that you can not only modify the identifier but also the information itself.

Notify types

Default

This setting disables translation of any kind. The information is passed 1:1.

Not-Operation

If a value of zero arrives, a 1 will be passed on. In all other cases the new information will have a value of 0. This is similar to a logical NOT operation, note however that TRUE is not only represented by 1, but also by all other values unequal zero.

Increment

With this setting the information will be incremented by 1. This is for example useful to link a cycle gadget with a date gadget. The cycle gadget returns for its first entry (in our case, the months of the year) a 0 and for the second a 1. The date gadget however does not understand a 0 when its month is set, thus the link object should add 1 to the cycle gadgets information.

Decrement

Similar to Increment, however the information will be decremented by 1.

2^x

This setting will calculate the exponent of an information. The following formula is used:

<New information> = 2 to the power of <old information>

This setting is eg. useful to adjust the color depth.

True

No matter, which information arrives, the link object will always pass on a 1.

False

The new information will always be a 0.

Important!

The converting process does always occur and is controlled through the lower two cycle gadgets within the Notify editor. The type setting does not affect this at all.

If you wish to create a complex menu using toggle gadgets (where only one gadget is active at the same time), you should connect these with each other using link objects. In this case the type setting should be a NOT operation.

In order to make each menu option undeselectable you should set the "Simple Mode" flag for each Toggle object. This will force the Toggle gadget to remained in a "pushed" mode.

1.16 StormWIZARD.guide/ST_Project

Programing

Although the variety of possibilities StormWIZARD offers to integrate automatic operations already during the development of the user interface, this is yet not enough to create complete program.

This chapter will show you how to use the resources created by StormWIZARD in your own programs. All basics required to do this will be explained using simple C examples. Additionally, the example programs in the "StormWIZARD/Example-Source" drawer may help to answer all of your questions.

Header files

Every object you created using StormWIZARD can be assigned a name. This name will e.g. be used to set and read object attributes. In turn, every name is assigned an ID number which will be saved when a Wizard resource is written into a header file. By default, the C syntax will be used. As Pascal, Basic and E programs have a different definition syntax you can of course adjust this setting.

The defaults are defined using the icon attributes:

HEADER_FORMAT

HEADER_SUFFIX

where HEADER_FORMAT describes the format of the definition and HEADER_SUFFIX specifies the header file's suffix.

To show an example, the OK_BUTTON is assigned the ID-Number 42. The header file's definition will be generated like this:

#define OK_BOTTON 0x2A

The tool type should then have the following format:

HEADER_FORMAT=#define %s 0x%lx

You'll recognise the C-style syntax as used for output functions, e.g. for "printf" to describe the type of formatting which should occur.

To create a correct output file for BlitzBasic the tool types should be changed as follows:

HEADER_FORMAT=%s %ld

HEADER_SUFFIX=.bbh

The header file will then look like this:

OK_BUTTON 42

BASE PROGRAMS

Opening the library

In order to access the library's functions, you must of course open it first. This is done in the Amiga-typic way:

struct Library *WizardBase; // define the library's base address

.

.

.

if((WizardBase = OpenLibrary("wizard.library",0L)))

{

// Library has been successfully opened

}

else

// Error handling!

If you're programming using StormC, the Storm.lib link library and the StormC startup code, the library will be automatically opened for you.

Closing the library

When your program terminates you should close the library so that the memory it used can be freed if necessary.

This could look like this:

```
if( WizardBase)
CloseLibrary( WizardBase);
// No Wizard functions any more!!!
```

Loading a Wizard file

In order to maker a layout description accessible you should use the WZ_OpenSurface() function. Here's an example:

```
APTR MySurface;
if( (MySurface = WZ_OpenSurface("manager.wizard",0L,TAG_DONE)))
{
// File was successfully loaded
}
else // Error handling!
```

The memory address the Wizard file was loaded to will be returned. If an error occurred, the function will return zero.

Localisation

If you wish to use StormWizard's Localisation features you need to specify the address of the appropriate Locale catalogue, too.

This means that before you can open the layout description you need to load the catalogue using locale.library functions.

Keep in mind that herefore locale.library must be opened, too.

An example:

```
APTR MySurface;
struct Catalog *MyCatalog;
if (( MyCatalog = OpenCatalog( NULL, "program.catalog", TAG_DONE)))
{
if (( MySurface = WZ_OpenSurface( "program.wizard", NULL,
SFH_Catalog, MyCatalog, TAG_DONE)))
{
.
.
WZ_CloseSurface( MySurface);
}
CloseCatalog( MyCatalog);
}
```

You do not necessarily need to check whether the catalogue file could be loaded or not.

Wizard files in memory

The Zero parameter passed above is a pointer to a memory address. If the ".wizard" file has already been loaded into memory (eg. by binary include) you should specify its address and pass a Zero for the name of the catalogue-

Example:

MySurface=WZ_OpenSurface(0L, SurfaceAddress, TAG_DONE);

Creating windows

Now we finally want to create a window. However we'll first need to allocate a so-called WindowHandle. The function is called as shown below:

struct WizardWindowHandle *MyWinHandle;

MyWinHandle=WZ_AllocWindowHandle(MyScreen,

sizeof(MyWinExtension),

MySurface,

TAG_DONE);

MyScreen is a pointer to the screen you wish to open the window on.

At the same time you may specify the address of a private structure which is automatically allocated by this function. This is useful if you need to manage your own information belonging to the window. The address of the structure allocated can then be obtained from the WizardWindowHandle structure.

If you do not wish to use this possibility, specify a Zero here.

MySurface is the handle returned by the previous call to WZ_OpenSurface.

Last not least you can also pass Tag arguments. Currently the only supported tag is WWH_StackSize. This tag describes the size of the window-specific stack which is used for the layout process. This stack is automatically allocated when the function is called.

Freeing windows

The WindowHandle should be released as soon as possible if you don't need it any longer. This will automatically release all objects attached to the WindowHandle. This also affects possibly opened windows!

WZ_FreeWindowHandle(MyWinHandle);

Having only the WindowHandle does not give you any advantage, so there's a function to create all objects belonging to a window. This affects gadgets, menus, notify and other objects.

Example:

#define MY_WINDOW_ID 1

#define MY_WINDOW_GADGETS 80

struct NewWindow *MyNewWindow;

struct Gadget *MyGadgets[MY_WINDOW_GADGETS];

MyNewWindow = WZ_CreateWindowObj(MyWinHandle,

MY_WINDOW_ID,

WWH_GadgetArray, MyGadgets,

WWH_GadgetArraySize,

sizeof(MyGadgets),

TAG_DONE);

We do not want to explain MyWinHandle any longer, however MY_WINDOW_ID needs an explanation: this is the identification number assigned by StormWIZARD when you save the layout.

You can find this ID along with the object name specified in the Attribute editor in the Include file created when you save the Wizard file.

The tag list should at least contain the WWH_GadgetArray tag which is used to specify the address of your gadget list - you'd also like to access the gadgets created, wouldn't you?

The array's size should be chosen carefully so that it is able to host all gadgets. To make sure that the gadget array is only accepted when it is big enough you should also specify its size in bytes using the WWH_GadgetArraySize tag.

If all objects could be created properly you'll receive the address of an already initialised NewWindow structure.

You may modify this structure to your needs and need it for the call of the following function:

struct Window *MyWindow;

MyWindow=WZ_OpenWindow(MyWinHandle, MyNewWindow,

WA_AutoAdjust ,TRUE, TAG_DONE);

You already know the MyWinHandle and MyNewWindow parameters from the previous functions. A new parameter is the tag list where all tags valid for the OpenWindowTagList() function can be specified.

It is recommended to set the WA_AutoAdjust tag to TRUE so that the window can be resized if necessary.

Analogously, there is a function to close the window:

WZ_CloseWindow(MyWinhandle);

You can open and close windows as much as you like to!

Receiving messages

A window is pretty useless if you don't listen for IDCMP messages at the same time. Use Wait() and WaitPort() for this purpose.

Fetch the message as usual using GetMsg() from the window's UserPort. If you also wish to use the keyboard support provided by the Wizard classes take a look at the following excerpt:

switch(msg->Class)

{

case IDCMP_VANILLAKEY:

WZ_GadgetKey(MyWinHandle, msg->Code, msg->Qualifier, TAG_DONE);

break;

}

Note that if a gadget is responsible for this keypress it is caused to generate a IDCMP_IDCMPUPDATE on its own. Notify objects will receive this message, too.

If the gadget is an Integer or a String gadget it will be activated without sending a message. The function also returns whether a gadget is responsible at all.

In our example however we did not take care of this.

You should pass all ASCII key messages to the WZ_GadgetKey() function.

If the function returns FALSE it could find no appropriate gadget for the keypress. In this case you can check for your own keycodes.

Gadget help

Starting with OS 3.0 the Amiga knows the IDCMP_GADGETHELP flag. It is often used to display a help text for the user. StormWizard provides a special function to assign a help text to each function. Here is an example:

STRPTR Help;

Help=WZ_GadgetHelp(MyWinHandle, msg->IAddress);

It is possible that the function returns zero instead of an address, thus check its result. There's a similar function for menus for the case that IDCMP_MENUHELP messages arrive:

STRPTR Help;

Help=WZ_MenuHelp(MyWinHandle, msg->Code);

Unfortunately the system provides this function only in V39 and newer versions. Often however you'll want your program to run already under V37 (OS 2.0/2.1). For this reason the Wizard.library has a replacement function which simulates the GadgetHelp message. This requires watching mouse movement.

```
STRPTR Help;

APTR HelpIAddress;

struct WizardWindowHandle *HelpWinHandle;

.

.

.

case IDCMP_MOUSEMOVE:

if (WZ_GadgetHelp( MyWinHandle, &HelpWinHandle, &HelpIAddress,

msg->MouseX, msg->MouseY, 0))

{

Help=WZ_GadgetHelp( HelpWinHandle, HelpIAddress);

}

break;
```

MyWinHandle is the window receiving the IDCMP_MOUSEMOVE message.

HelpWinHandle contains a pointer to a WizardWindowHandle when a GadgetHelp message arrives.

HelpIAddress then contains the same value as IntuiMessage->IAddress.

Of course we also need the mouse position and the flags influencing the function's behaviour.

If the WGHF_IgnoreOS flag is set the GadgetHelp function is simulated regardless of the Operating System the program runs on. Otherwise the system's function will be used if the program runs under AmigaOS V39.

The WGHF_FullControl flag is currently unused.

Locking windows

Often you'll need to lock an opened window against user input. For this purpose the library contains a function which requires the window's WizardWindowHandle. You can call this function multiple times.

WZ_LockWindow(MyWinHandle); // Window is locked now

To unlock the window for user input the following function requires the same parameter:

WZ_UnlockWindow(MyWinHandle); // Window accepts user input again

Note that to completely unlock the window you'll need to call the function as much as you called WZ_LockWindow().

If you're working with more than a single window you can lock all windows at once with another function.

Here the parameter is the address returned by WZ_OpenSurface(). Mind the appended "s"!

WZ_LockWindows(MySurface);

To unlock the windows use:

WZ_UnlockWindows(MySurace);

SPECIAL PROGRAMMING TECHNIQUES

As we already said before gadgets need some parameters to be initialised already before you open the window. Otherwise accesses to memory address 0 may occur.

ListView and MultiListView

These gadgets need the address of the View list structure. Use the WLISTVIEWA_List tag for this purpose!

Hierarchy

This gadget also needs the list's address, however the tag responsible is called WHIERARCYA_List.

Args

As you know this class can format 10 different parameters according to a format specification. You'll need to know which parameters you want to set. Use the tags from WARGSA_Arg0 to WARGS_Arg9. It is recommended to set all parameters when calling the function. If you want the object to display numeric values only you don't necessarily need to set the attributes.

An example:

Your window contains an Args gadget with the format string "Year: %ld Week day: %s".

As you can see the second parameter requires a string. For this reason you need to pass the gadget at least the second parameter before you open the window:

char MyString[256] = "Monday";

SetGadgetAttrs(argsgadget, 0L, 0L, WARGSA_Arg0, MyString, TAG_DONE);

You might have noticed that we passed a 0 as Window argument as the window does not yet exist. Thus you must set the attributes immediately after having created the gadgets using WZ_CreateWindowObj()!

After you have set these attributes you can open the window without any problems. All gadgets will render themselves and the output formatting can be performed safely.

Requester functions

WZ_EasyRequestArgs()'s probably most important advantage is that it supports the keyboard and thus saves the user from having to switch between mouse and keyboard all the time.

#define MY_REQUESTER_ID 1

LONG result;

WZ_LockWindow(MyWinHandle);

result=WZ_EasyRequestArgs(MySurface, MyWindow, MY_REQUESTER_ID, &Args);

WZ_UnlockWindow(MyWinHandle);

As regards parameters, this function expects the address returned by von WZ_OpenSurface() followed by the address of the window, which is used to determine the screen in whose upper left edge the requester will open. Then you should pass the identification number you entered in StormWizard. The last argument is a pointer to a memory block containing data for the formatting string - this enables you dynamically determine the text displayed. The function returns the same values as the system's EasyRequestArgs() function. Two more things to notice are that unlike the system function StormWizard's version does not check for disk changes and similar events and that WZ_EasyRequestArgs does not lock the window: you'll need to do this on your own using WZ_LockWindow() and similar functions.

Next here's an important function to write your own requesters:

ULONG Myidcmp;

struct EasyStruct MyEasystruct,*easy;

easy=WZ_InitEasyStruct(MySurface, &MyEasystruct, &Myidcmp,

MY_REQUESTER_ID, sizeof(MyEasystruct));

This function initialises an EasyRequester structure. It returns the structure's address you passed if no errors occurred and zero otherwise.

Programming ListViews

Initialise a WizardNode structure using the WZ_InitNode() function.

Pass a pointer to the WizardNode, the number of entries you wish to create and a series of tags as arguments.

As soon as you've initialised the WizardNode you should fill it with data using the function WZ_InitNodeEntry() function which of course requires the WizardNode's address, followed by the entry to be modified and a tag list containing your data.

How do I create a WizardNode for a List- or Multilistview?

You'll need to use a WizardDefaultNode! This is a WizardNode with a single entry which is initialised as follows:

WZ_InitNode(&WizardDefaultNode, 1, WNODEA_Flags, 0, TAG_DONE);

WZ_InitNodeEntry(&WizardDefaultNode, 0, WENTRY_Type, WNE_TEXT,

WENTRYA_TextPen, WZRD_TEXTPEN,

WENTRYA_TextSPen, WZRD_FILLTEXTPEN,

WENTRYA_TextStyle, FSF_BOLD,

WENTRYA_TextSStyle, FSF_BOLD,

WENTRYA_TextString, NodeString,

TAG_DONE);

Here's an example for a Hierarchy gadget:

WZ_InitNode(&WizardDefaultNode, 1, WNODEA_Flags, WNF_TREE|WNF_AUTOMATIC,

TAG_DONE);

WZ_InitNodeEntry(&WizardNode, 0, WENTRY_Type, WNE_TREE,

WENTRYA_TreeParentNode, &ParentNode,

WENTRYA_TreeChilds, 0,

WENTRYA_TreePen, WZRD_TEXTPEN,

WENTRYA_TreeSPen, WZRD_FILLTEXTPEN,

WENTRYA_TreeString, NodeString,

TAG_DONE);

It is strongly recommended that you take a look at the three example programs as these show you in detail how these functions are used.

Unfortunately V37 of the library does not yet allow you to create your own Nodes. To remain compatible to further library versions it is very important that you use these functions.

TIPS, TRICKS UND COMPATIBILITY ISSUES

Modifying a Listview list

If you wish to modify a list which is already used by a Listview gadget you'll need to temporarily detach the list by resetting the gadget's attributes using the WLISTVIEWA_List tag together with 0 as address.

As you soon as you've completed your changes reattach the list (using WLISTVIEWA_List and the list's address).

Of course this also applies for the Multilistview and Hierarchy classes.

Stack size

You should always keep an eye on the stack size as many operations are performed recursively. For example, WGA_MinWidth and WGA_MinHeight requests addressed to group gadgets need your special attention as their calculation routines work recursively. These tags can be requested from all Wizard gadgets.

Alien gadgets

If you wish to use your own "alien" gadgets in the window you should use the WWH_PreviousGadget tag.

It is recommended to insert all alien BOOPSI objects in front of the Wizard gadgets.

Note

The best example showing how to include alien BOOPSI gadgets can be found in the "Palette.c" demo.

Alien BOOSPI objects should be created after the WindowHandle has been obtained as it is then possible to use the window's DrawInfo structure. Also, you can link your private BOOPSI objects into the object list. This will automatically release them when WZ_FreeWindowHandle() is called.

The WindowHandle is a read-only structure - do never try to write into it!

Calling WZ_FreeWindowHandle() will releases all resources belonging to a window.

WZ_CloseSurface() removes all remaining WindowHandles. Logically, this closes all remaining windows.

Be careful not to use SetAttrsA() for Wizard gadgets as this can cause the system to crash.

Never directly access a class's internal structures, even if you discover that an attribute is saved at a particular address - the location will change for sure. Information exchange between gadgets must be performed using GetAttr() and SetGadgetAttrsA(). These functions are the only way to guarantee that you programs will work safe in the future.

Autodocs

The attributes defined in the Autodocs for particular classes must only be used as is written there.

The attributes can be found in the Autodocs under the description of the WZ_NewObject function.

Currently you can also test for the tags sent along with a Notify using GetAttr(). If the tag carries a "N" this will be no longer available in the next version, which is already under development. You must take care of each tag's attributes!

Example:

WGROUPA_ActivePage -- (V37) [CSGN], UWORD

The square brackets following the tag have the following meaning:

C The tag can be used for Create and is also often required.

S The tag can be specified in a SetGadgetAttrsA() call.

G The tag's status can be checked using GetAttr().

N This tag is passed along with a message.

Saving window positions

When the window is closed using WZ_CloseWindow() the window positions and dimensions are saved into the NewWindow structure whose address you've received from calling WZ_CreateWindowObj().

This is performed automatically. Keep track of the NewWindow structure in your private structures.

Private data structures

The function which allocates the WindowHandle can automatically allocate a private structure for usage by your program. You could e.g. use this possibility to create a MinNode structure which links your WindowHandles if your program has to deal with multiple windows. It is also a good idea to set the UserData field in Intuition's window structure to the address of the WindowHandle. In any case, never allocate a WindowHandle using AllocVec() or similar functions.

Style Guide

Keep in mind that your GUI does not become too coloured. An interface which looks obviously overloaded rather makes it difficult for the user to use than fulfilling its real purpose, making the program easier to use.

Follow the Style Guide's rules as regards your GUI's look.

To keep your window tidy you should use the "Paging" feature provided by the Group gadgets. Mind the limit of 32 pages.

If you need more than 32 pages you'll have to nest multiple group gadgets into each other.

Links should always appear on the same page. If you don't use "Paging" at all this will of course be no problem for you.

Help texts

Use the help strings, which are available for all Wizard objects, intensively! They will also appear in the ".cd" catalogue description file when the final code is created. Hereby, when you translate these strings into another language, your online help function will automatically be localised, too.

Use the library as intensive as possible - so you'll have as less work as possible with the GUI.

Object linking

StormWizard offers you a lot of Notify links, especially concerning the selectibility of gadgets. Use this feature as the GUI can perform this on its own without any line of custom code. Notify links also work if you don't fetch messages from the message port.

Often it is possible to combine gadgets of the same or a different class to a visibly new object. Notify links also allow to manipulate messages exchanged under the gadgets. For example, this allows to combine a series of Toggle gadgets to a single object.

Your program does not need to be recompiled if you change its graphical user interface.

The only exception: if you're using Menus the menu code is determined by the entry's position so that modifications in this area require recompiling.

Screen manager

Starting with OS 2.0 the Amiga knows Public Screens. Window-orientated applications should use this feature to give the user full control over the screen's resolution, colour depth and dimensions. Also this allows the user to use his monitor's full capabilities. To manage these screens we've written the StormScreenManager utility which features an Arexx port understanding the commands "OPEN" and "CLOSE". Any screen opened via Arexx will be automatically closed as soon as the last screen lock has been released. It is sufficient to obtain the screen using "OPEN NAME". If the screen could be opened you should obtain its address using LockPubScreen(). If this function returns zero the screen could not be opened and you should use the Workbench screen instead.

ARexx

Arexx messages can be sent to StormWizard using the rexxsyslib.library and PutMsg(). Please don't forget to initialise the rm_LibBase with the library's base address.

Dock mode

The dock mode provided by the Group gadgets will become pixel-orientated in the feature. Always link a Group gadget with a Scroller object in both directions. The issue which object is rendered visible and which not is handled internally!

As a window object's minimum dimensions are already determined when you create the object it will be from this moment on referenced from this location. Even when you're working with fixed fonts the minimum dimensions must not be considered constant. To obtain these use the WGA_MinHeight and WGA_MinWidth tags every time when the window has been created!

When Group gadgets work in Paging mode and thus provide their own gadgets the background colour must be set to zero.

Though a ListView must always be linked with a vertical Scroller you must not obtain any values from the Scroller using GetAttr(). The scrolling will perform pixel-orientated in a future version of the library and GetAttr() would then return false values.

1.17 StormWIZARD.guide/ST_Project

Reference

So, you've gone through all exercises, learned about StormWizard's layout techniques and already created your first programs all you still need is this part of the manual: the reference!

This chapter serves as a reference guide for creating and modifying your layouts. You'll find short explanations for all attribute editors and some helpful hints.

The Window editor

Add window

Using this option you can add a new window. The window's attribute editor will be opened.

Edit

Depending on the currently active element in the hierachy list this button will take you to:

* the Window attribute editor if you selected a window.

* the Gadget editor if you selected "Gadgets",

* the Menu editor if you selected "Menu".

Open window

This button will open the selected window so that you can verify its contents.

Close window

This will close the opened window. If your window has a Close gadget and it is able to receive appropiate messages it can also be closed by a click on this Close gadget.

Up/Down

If you're working on huge projects with many windows it is useful to move windows with many objects to the start of the list as they will open much faster at this position. You should also place windows here which you open very often.

Hierarchy Listview

Double-clicking an element in the list will take you to:

* the Window attribute editor if you clicked on a window,

* the Gadget editor if you clicked on "Gadgets",

* the Menu editor if you clicked on "Menu".

Project menu

The Window editor's menus

New

Removes all objects from all of the program's editors. Fonts, Requests and Windows will be released including their subobjects and will not be available any longer. You'll be asked to confirm this option.

Open

To edit a ".wizard" file you first need to load it. All objects saved in such a file are then available in the appropiate editors. Already existing objects will be overwritten.

Save

This menu option will save all changes back to the currently open file. If this is a new project a file requester will appear asking you to enter a path and file name.

Save as

Saves the entire project into a file. Depending on the program's toll types a second file will be created containing all object names. This option will always open a file requester.

About

This will open a window with a Copyright notice. Furthermore you'll be able to see the current window's gadget buffer's size. The specified size will be assigned to the window's object name. As the object name is used to open the window the gadget constant's name will receive a _GADGETS suffix.

If you eg. name your window WIN_MAIN the generated Include file will also define WIN_MAIN_GADGETS. You should define the gadget buffer in your program as follows:

struct Gadget * MyGadgets(WIN_MAIN_GADGETS);

Quit

Quits the program. All created objects will be lost (unless you saved them into a file). A confirmation requester will appear.

Edit menu

Cut

Using this menu option you can "cut" a window. The appropriate entry will be removed from the window editor. The window is now available in the clipboard and can be pasted at any time.

If you cut a window all of its objects will be cut, too, except for alternative fonts within gadgets. In this case a requester will appear asking you to confirm to cut the window without these fonts.

Copy

The selected entry will be copied and unlike "Cut" not removed.

Paste

This menu option is used to paste a window from the program's clipboard. The new window object will be inserted below the currently highlighted object.

When you paste an object its object name will be cleared. You should assign it a new name by double-clicking the new entry.

Delete

This will completely remove the selected window object including all of its subobjects. Be careful, this option will not ask you to confirm!

Tools menu

Request editor

The "wizard.library" also eases the programming of simple requesters. Selecting this entry will take you the Request editor.

Version editor

This entry allows you enter a version number for your project which will be saved into the project's FVER chunk. Don't enter the project name and date as StormWizard will do this for you. Also whitespaces are not allowed.

The Window attribute editor

Object name

This entry plays an important role in your program. The object name is used to distinguish the objects from each other and must thus be a unique name. When you save your project a seperate file containing the object names is created which you can include in your program.

Left and TopEdge

Using these fields you can specify the position of the upper left edge of your window. When you preview your layout the window will open at this position.

When you call wizard.library's WZ_CreateWindowObj() function from your program these values will be placed into the NewWindow structure.

Width and Height

Similar to Left and TopEdge these values will be written into the NewWindow structure when you preview or open the window from within your program. If however the minimum dimension is bigger than the values specified here the library will automatically correct these values. Do not specify negative values.

Window title

Whatever you enter here will appear in your window's title bar later on.

Screen title

The text specified here will become the current screen's title when the window is selected. If this field is left empty, the screen will keep its own title.

Important!

StormWizard renders its help strings into the screen's title bar so that the screen's title will not be visible in these cases!

Help text

As the library provides its own help system you should specify a help string for the window. If the mouse is positioned within your window, but not on top of a gadget, WZ_GadgetHelp() returns this string!

Window flags

The window flags determine which system gadgets should be displayed and how the window should behave when its contents are redrawn.

SizeGadget System gadget to in- or decrease the window’s size.

DragBar Makes the window moveable.

DepthGadget The window can be clicked into the fore- or background.

CloseGadget Used to close the window (generates IDCMP_CLOSEWINDOW messages).

SizeBRight The Size gadget will be positioned in the window’s right border..

SizeBBottom The Size gadget will be positioned in the window’s lower border:

SimpleRefresh All window redrawing will be done manually by the program. In contrast to "SmartRefresh" (which is the default attribute) this attribute puts all control over the window’s contents into the program’s hands. "SimpleRefresh" windows use more CPU time but require less memory than "SmartRefresh" windows.

ReportMouse Reports the mouse pointer’s current position to the program.

NoCareRefresh Disables the sending of the IDCMP_REFRESHWINDOW message. The system will take care of all window refresh operations. This flag should always be set under AmigaOS 2.0!

Borderless This will disable the rendering of the window’s borders.

Activate Upon opening the window will be automatically activated.

RMBTrap Enables the IDCMP_MOUSEBUTTONS message for the right mouse button.

NewLock All menus should be rendered using black text on white background. If this flag is disabled the menus will be drawn is Amiga OS 2.0 style.

Important!

In order to receive MouseMove messages for your windows you must set the ReportMouse flag.

IDCMP flags

These are the same as Intuition’s flags.

Important!

Under OS 2.0 StormWizard and the wizard.library’s help system requires the MouseMove flag. Starting with OS 3.0 StormWizard uses GadgetHelp messages.

Ok

Closes the window and applies the changes performed.

Cancel

All changes will be cancelled and the window will be closed.

The Font editor

User name

Directly behind this label you’ll find a Vector-Popup gadget which is used to select the font to be edited. If you don’t like the name the font uses any longer you can enter a custom name in the string gadget next to it. The string gadget must be confirmed using <Return>.

Proportional font

This CheckBox restricts the font selection: if the box is checked you’ll also be able to select proportional fonts, where each char has its own width.

Font

If you wish to change the font and/or its size this VectorButton gadget will open a font requester.

Ok

Closes the window and applies the changes performed.

Cancel

All changes will be cancelled and the window will be closed.

Font editor menu

Project menu

Close

Closes the font editor and returns to the window editor. All changes will be applied.

The Requester editor

Display

To display your created requester simply click this gadget once.

Add

Adds a new requester and opens the requester attribute editor. The new requester will be inserted below the currently highlighted entry.

Edit

This will open the requester attribute editor where you can modify the requester's attributes.

Up and Down

Using these buttons you can move an entry up and down. These gadgets should help you to group the requesters the way you want.

Hierarchy Listview

Double-clicking an entry opens the requester attribute editor.

Project menu

The Requester editor's menus

Close

Closes the requester editor and returns to the window editor. All changes will be applied.

Edit menu

Cut

Cuts the currently highlighted requester into the program's internal clipboard. The requester object will be removed from the list.

Copy

Copies the currently highlighted requester to the clipboard. This will in contrast to "Cut" not remove the requester object.

Paste

Inserts the requester currently in the program's clipboard below the currently highlighted entry.

Delete

Removes the highlighted requester without further confirmation requests!

Renumber

Every gadget object has its own identification number. Sometimes it is necessary to reassign these numbers. This operation will affect all object within the list which belong to the current window.

The Requester attribute editor

Object name

This entry plays an important role in your program. The object name is used to distinguish the objects from each other and must thus be a unique name. When you save your project a seperate file containing the object names is created which you can include in your program.

Title

Whatever you enter here will automatically appear in the requester's title bar.

Gadgets

A requester can only be closed using its gadgets as there is no clear definition whether a click on a close gadget should be interpreted as a positive or negative response. For this reason, StormWizard always inserts at least one gadget automatically, even if you don't specify a text in the "Gadget" field. If you wish to use multiple gadgets specify the texts with a vertical bar. Example: "OK|Cancel".

Format

This specifies the text to appear in the requester. You can use all percent-sign tokens known from the system (eg. %s). The actual contents are passed in the program itself when you call WZ_EasyRequestArgs().

Ok

Closes the window and applies the changes performed.

Cancel

All changes will be cancelled and the window will be closed.

The Menu editor

Redraw

Clicking "Redraw" redraws the Preview window based on the changes performed so that you can directly see and test them.

Add menu

To create a new menu title or item click on "Add menu". You will at first need to define a menu title using the Menu Attribute editor which will appear.

Up/Down

Using these keys you can move the selected entry up and down. Using these technique you can attach and detach entries to and from other entries.

If you move a menu title below another menu title it will automatically change to a menu item.

Double-clicking a menu entry will open the Menu Attribute editor.

Hierarchy Listview

Double-clicking the first entry will open the Menu Attribute editor while clicking on all other entries will open the Menu Entry Attribute editor. In both cases you'll be given the chance to change the object's attributes.

The Menu editor's menus

Project menu

Close

Closes the Menu editor and returns to the Window editor. If you performed any changes to the Menus, these will be finally applied.

Edit menu

Cut

The selected menu will be removed from the list and stored in the program's internal clipboard. You can insert it at a new location using the "Paste" command. If the menu entry contains submenus, this operation of course also affects these submenus.

Copy

Selecting "Copy" will create a copy of the entry in the program's clipboard.

Paste

Inserts the menu which is currently in the program's clipboard behind the selected menu entry in the list. If the entry inserted is a menu entry or a submenu entry you can move it to the desired position using the "Up" and "Down" gadgets.

Delete

This menu entry will completely delete the menu entry including its sub menus.

WARNING!

This operation can not be undone!

The Menu Attribute editor

Objectname

This entry plays an important role in your program. The object name is used to distinguish the objects from each other and must thus be a unique name. When you save your project a seperate file containing the object names is created which you can include in your program.

Name

Specify the menu's title here.

Help text

Starting with OS 3.0 the system provides a help function which is available through the Help key. Specify the help text in this string gadget if you wish to make such a help text available through WZ_MenuHelp().

Configuration

Every StormWizard object has its own string which can be fetched through the WZ_GadgetConfig() function. The function specified here is for your free use. You may save your own configurations or other things important for your program.

Disabled

If you wish to render the menu title to be unavailable set this flag. The menu title will appear "ghosted".

Ok

Closes the Menu Attribute editor and applies all changes performed.

Cancel

The window will be closed and all changes will be cancelled.

The Menu Entry editor

Object name

This entry plays an important role in your program. The object name is used to distinguish the objects from each other and must thus be a unique name. When you save your project a seperate file containing the object names is created which you can include in your program.

Name

Specify the title of the Menu here.

Help text

Starting with OS 3.0 the system provides a help function which is available through the Help key. Specify the help text in this string gadget if you wish to make such a help text available through WZ_MenuHelp().

Configuration

Every StormWizard object has its own string which can be fetched through the WZ_GadgetConfig() function. The function specified here is for your free use. You may save your own configurations or other things important for your program.

Shortcut

The system also allows you to select a menu entry using your keyboard. Normally, the qualifier used is the right Amiga key. If the user hits the key specified here together with the right Amiga key, your program will receive the same IDCMP_MENUPICK message you'd receive if the user had selected the menu entry using the mouse.

Mutual-Exclusion

This field should only be used if the "CheckIt" flag is set. If you want the system to uncheck other menu entries when this entry is selected you can specify a 32-bit number here which describes the entries to be unchecked.

Example:

The illustration to the right shows a menu we'll be referring to in this example.

All menu entries have their "CheckIt" and "MenuToggle" flags set. You will notice that all entries can be checked simultanously whereas we want all other entries to be unchecked when an entry is selected.

For this purpose we fill the menu entry's Mutual-Exclusion field with a bitmask which is generated as follows:

If you want a single entry to be unchecked when an entry is selected it is sufficient to specify the appropiate bit in the Mutual-Exclusion field.

If you want multiple entries to be unchecked at the same time you need to add the bit values of these entries.

In our example we want all other checkmarks to be removed when any other entry is selected. To achieve this, we assign the menu entries the following Mutual-Exclusion values:

1. / First entry -> 30 (2+4+8+16)

2. / Second entry -> 29 (1+4+8+16)

3. / Third entry -> 27 (1+2+8+16)

4. / Fourth entry -> 23 (1+2+4+16)

5. / Fifth entry -> 15 (1+2+4+8)

Disabled

If you wish this menu entry to be unselectable, enable this flag and the entry will appear "ghosted".

Command string

If you wish to assign this menu entry a special key combination different than the right Amiga shortcuts, check this flag and enter the string in the "ShortKey" field. In this case the menu will show the specified string instead of the shortcut. It is then however left up to you to evaluate the keypresses.

CheckIt

If you wish the menu entry to be checkable, enable this flag.

Checked

If you've enabled "CheckIt" and you want this item to be already selected when the menu is accessed, enable this flag.

MenuToggle

The "MenuToggle" should always be used together with the "CheckIt" flag. Otherwise the entry will be selectable and appear checked, the checkmark will however remain unremoveable.

Ok

Closes the window and applies all changes performed.

Cancel

The window will be closed and all changes will be cancelled.

The Gadget editor

Redraw

Clicking "Redraw" redraws the Preview window based on the changes performed. The window's position and dimensions will be saved in the window object.

Add object

The "Add object" button is a Text-Popup gadget. When you click on it you'll be shown a list with possible selections. After you've chosen such an object its appropiate attribute editor will be opened. You'll have to confirm the editor in order for the hierarchy list to show the new object.

Edit

Clicking on "Edit" will open the class-specific attribute editor. Alternatively, you can also double-click on the entry itself.

Show position

If the Preview window is opened, clicking on "Show position" will show you the position and the dimensions of the selected object within the Preview window.

Up/Down

Using these keys you can move the selected entry up and down. Using these technique you can attach and detach gadgets to and from other gadgets.

H<->V

Some classes only differ in their vertical or horizontal alignment. The "H<->V" button allows you to quickly change this attribute. This currently only works for group gadgets and proportional gadgets (sliders).

Notify editor

In some events some classes send messages which can be received and interpreted by other classes. Often it's then the case that other objects within the window need to adjust their properties. To allow such events to be programmed with few mouse clicks, a powerful Notify system has been developed.

As these Notify links always require a "transmitter object", clicking this button will assume that the currently selected gadget is this "transmitter object". You can then specify as many "receiver objects" as you like to.

Hierarchy Listview

A double-click on a gadget entry always opens the attribute editor responsible for this class. Following this rule, a double-click on ie. a button object will open the Button Attribute editor.

The Gadget editor's menus

Project menu

Close

Closes the Gadget editor and returns to the Window editor. If you performed any changes to the gadgets, these will be finally applied.

Edit menu

Cut

The selected gadget will be removed from the list and stored in the program's internal clipboard. You can insert it at a new location using the "Paste" command. If the object cut is a group gadget, this operation of course also affects its subobjects.

Copy

Selecting "Copy" will create a copy of the entry in the program's clipboard.

Paste

Inserts the gadget which is currently in the program's clipboard behind the selected gadget entry in the list. If the object inserted is a group gadget all of its subobjects will be inserted, too, while assigning them new gadget IDs. You'll have to specify a new object name on your own. Links and Notifies are restored as far as possible!

Delete

If you wish to delete a gadget entry, including its members it possibly has, use this function.

WARNING!

This operation can not be undone!

Renumber

As was already said above, each gadget object has its own unique identification number. Sometimes it may be necessary to reassign or to reorder these numbers. This operation will affect all objects within the list which belongs to the current window.

General attributes (HGroup- and VGroup attributes)

In general, the object's attribute editors all look the same which is why we'll use the HGroup-VGroup attribute editor to show what settings can be configured for all objects. We'll then only discuss the special configuration possibilities for some objects.

Objectname

This entry plays an important role in your program. The object name is used to distinguish the objects from each other and must thus be a unique name. When you save your project a seperate file containing the object names is created which you can include in your program.

Page

If your gadget is member of a group gadget, this number determines the page on which it will be displayed. As group gadgets can only manage up to 32 pages, this number must be within the range from 0 to 31. If the appropiate group gadget has only 2 pages, the number must be either 0 or 1.

Help text

Specify the help text here, which should be sent as a message when the mouse pointer is positioned on top of the gadget. This text can be queried using the library's WZ_GadgetHelp() function. Please note that the "GadgetHelp" flag must be set in order to enable the help function.

Configuration

Every StormWizard object has its own string which can be fetched through the WZ_GadgetConfig() function. The function specified here is for your free use. You may save your own configurations or other things important for your program.

Font

If you wish the group gadget to use its own font, you need to specify the font with a click on this Vector-Popup gadget. You'll be shown all fonts that have so far been defined.

Chose "New" from the menus if you wish to define a new font. The "New font" window will open where you can select your new font.

The selection "None", which is default for the gadget, will cause the library to use the screen's font.

Link

If a group gadget is operated in Dock mode it must be connected to a Scroller gadget. To inform the group gadget about movement of the slider's knob, you should create a Notify from the Scroller to the group gadget. This will let the objects exchange necessary information themselves.

Labels

Here you can specify a label text for the gadget group, which, if it can be displayed in a single line, will be displayed above of the group's members. If the label text consists of multiple lines, the group gadget will contain as many pages as the lines specified in the Listview. Also, the group gadget will render and manage a special gadget panel to switch between the single pages. If the entry is a vertical group gadget, this panel is rendered to the right and on top of each other. Horizontal group gadgets render the panel next to each other and above of its members.

Priority

This sets the group gadget's priority, which is used by the parent group gadget for its dimension and position calculations.

Whitespace proportions

A perfecnt value which determines the proportions between the objects and the whitespace dimensions. Refer to "Layout techniques".

Min. whitespace

The minimum size for the whitespace between two objects. Refer to "Layout techniques".

VBorder

Upper and lower whitespace. Refer to "Layout techniques".

HBorder

Whitespace to the left and right of the object. Refer to "Layout techniques".

BVOffset

Inner distance above and below of the gadget if a border is drawn. Refer to Layout techniques.

BHOffset

Inner distance to the left and the right of the gadget if a border is drawn. Refer to Layout techniques.

Max. pages

If you want the group gadget to reserve multiple pages for its members you'll need to specify the appropiate number of pages here. Do not specify a value higher than 31 as group gadgets only manage up to 32 pages. If you specified multiple headers on the "Labels" page this value is not used and should be set to 0.

Active page

If your group gadget consists of multiple pages you can use this setting to specify which page should be visible when the window is opened. If, however, the group gadget is not visible its subobjects are of course also not displayed.

Highlighting

The group-gadget will draw a pattern in the specified color into the whitespace areas between its members.

Border type

A group gadget offers you the possibility to draw a border around its members. The following illustruation shows the different variations possible.

GadgetHelp

Set this flag if you want a GadgetHelp message to be sent when the mouse is moved across the gadget.

Disabled

If you wish the gadget to be disabled when the window is opened you should set this flag. This does however only make sense if the group gadget contains of multiple pages and features a gadget panel to switch between these pages. The page's text will be rendered "ghosted" to indicate that it currently can not be selected.

EqualSize

If you wish all members of the group to have the same minimum size, set this flag. Keep in mind that the flag only affects the minimum dimensions! Refer to "Layout techniques" for further information.

Dock Mode

If a window is opened in Dock Mode, it can be resized as small as possible. The dimensions determined through the gadget the window contains are ignored. Refer to "Layout techniques" for further information.

Ok

Closes the window and applies all changes performed.

Cancel

The window will be closed and all changes will be cancelled.

Class-specific attributes

The following attributes are not valid in general for all objects and are only displayed in the Attribute editors for the appropiate class.

Name

Specify the object's text here. It will be displayed inside the object with a border around it.

If you wish the gadget to be selectable using the keyboard, put a "_" in front of the letter.

Example: "_Ok"

The following objects require a "Name":

Button

Toggle

RelHeight

The specified value is multiplied with a factor of two and is added to the font height. This is then the gadget's total height. Applicable values are 2 and 3. If you don't use a label you should specify a 0 here.

For objects of the "Checkbox" class , the font height is calculated from the linked label text.

"RelHeight" is only available for the following objects:

Button

String

Label

Checkbox

Mutual-Exclusion

Integer

H-V-Scroller

HSlider and VSlider

Listview

Toggle

Line

Colorfield

Args

Gauge

Cycle

Date

Text

MinWidth

The minimum width of the gadgets specified in pixels. Must be greater than zero.

The height of CheckBox gadgets is calculated as described under "RelHeight".

If you're working with images and the width of the image, added by 2 and multiplied by HBorder, is greater than the value specified here, the computed value will be used.

"MinWidth" is only available for the following objects:

String

CheckBox

Mutual-Exclusion

Integer

H-V-Scroller

HSlider and VSlider

Arrow

Listview

Line

Colorfield

Args

Gauge

Vectorbutton

Space

Image

Image-Button

Image-Toggle

Image-Popup

Palette

Vector-Popup

Hierarchy

MinHeight

The minimum height of the object in pixels.

If you specified a label for a line object, this label's text height is also used in the calculations.

Ist bei Image-Obekten der Wert aus der Hhe der Grafik + 2 * VBorder gr§er als der hier angegebene, gilt dieser.

If you're working with images and the height of the image, added by 2 and multiplied by VBorder, is greater than the value specified here, the computed value is used.

"MinWidth" is only available for the following objects:

H-V-Scroller

HSlider und VSlider

Arrow

Listview

Linie

Colorfield

Vectorbutton

Space

Image

Image-Button

Image-Toggle

Image-Popup

Palette

Vector-Popup

Hierarchy

String attributes

Init-String

This is the string the gadget originally contains when the window is opened.

MaxChars

The maximum length of the string in visible chars. Never specify a value greater than 255!

Label attributes

Labels

This is the text to be displayed by the label gadget. The vertical space between the lines is taken from "Linespace". If you wish to underline a single character within a line, put a "_" in front of it.

Bold

Causes the currently selected entry in the Labels Listview to be displayed in bold face.

Highlight

Causes the currently selected entry in the Labels Listview to be highlighted in the specified highlighting color.

Highlight color

This Vector-Popup gadget sets the highlighting color for the label texts.

LineSpace

This value is required to adjust the vertical space between the label gadgets.

Background

Label gadgets can optionally draw a coloured background which can be used ie. to emphasize a text. You should use the default values found in the Popup dialogue.

Text alignment

Of course you can also justify the text: either left-boundary, right-boundary or centered.

CheckBox attributes

Checked

Set this flag if you wish the box to be checked when the window is opened.

Integer attributes

Long

This is the long integer value the object will be initialized with when the window is opened.

MinLong

Signed 32-bit value which is used as the minimum value for the integer field. Lower entries will be rejected.

MaxLong

Signed 32-bit value which is used as the maximum value for the integer field. Higher entries will be rejected.

H-V-Scroller attributes

Top

The Top value describes the scroller's position in the range from 0 to "Total" minus "Visible".

Total

The total number of objects representated by the scroller.

Visible

"Visible" determines the visible number of entries in contrast to "Top" which specifies the first visible entry.

NewLook

To enable the 3D-Look for scroller gadgets set this flag. The 3D-Look is also available under AmigaOS 2.0.

HSliders and VSliders

Min

The minimum value for the slider.

Max

The maximum value for the slider.

Level

Current value of the slider.

New Look

To enable the 3D-Look available in OS 3.0 for scroller gadgets set this flag.

The look is also available under OS 2.0 as the "wizard.library" provides the appropiate objects.

Arrow attributes

Step

This value is sent by an Arrow gadget when it's sending a message due to an Intuitick. The first message sent is always a 1, regardless of this setting.

Type

Here you can choose one of four arrow types offered as vector graphics. If a notify exists, this setting also influences the direction the slider moves.

ListView attributes

Read only

This flag causes the Listview to be read-only, that is, no elements can be selected and the border with appear recessed.

Doubleclicks

If you want to allow double-clicks on entries be sure to set this flag. A double-click can be recognized by checking the received message's WLISTVIEWA_DoubleClick tag for the value TRUE.

Toggle attributes

Checked

If this flag is set, the toggle gadget will be displayed as if you already selected it before.

Simple Mode

If this flag is set, the Toggle gadget can not be deselected. This is useful if you combine a set of Toggle gadgets using Notifies to form a virtually complex object.

Line attributes

Label

This string will be displayed inside the horizontal line.

Line type

Here you can specify the type of the line. If a label is entered the type must be a horizontal line.

Colorfield attributes

Color

As the object is a Colorfield gadget it needs to know which color it should display. You're allowed to use the Wizard colors. The color will be shown inside the object.

Args attributes

Format

As this class was created to display different text displays you must specify the format string here. Please use only 32-bit placeholders. There are 10 different arguments available. Example: "I'm %ld years old".

Text alignment

Used to align the formatted text (left-boundary, right-boundary or centered).

Gauge attributes

Format

You can use a format string here to display ie. a procent display inside the gadget. Example: "%ld done".

Current

This is the value the gauge will be initialized with. The value must be an unsigned 32-bit integer and is interpreted relative to the total value.

Total

An unsigned 32-bit value which equals 100 percent (ie. if you perform an operation consisting of ten steps, you'd specify 10 here).

Vectorbutton attributes

Image

You can choose from three predefined image types: "File", "Drawer" or "Popup".

Date attributes

Day

Initializes the selected day. Depending on the month and year this value can be limited in different ways. Specify a 1 for the first day.

Month

Initializes the month to be displayed by the date gadget. Must be within the range from 1 to 12.

Year

Initializes the year to be displayed by the date gadget. Years before 1978 are not possible!

Image attributes

HBorder

Specifies the space left free to the left and right of the image. If you use a border you should not use a value of 0 as the border will then be drawn into the image itself.

VBorder

Specifies the space left free above and below of the image. If you use a border you should not use a value of 0 as the border will then be drawn into the image itself.

Background color

Specifies the color with which the free space is filled. It is recommended to use the default colors offered by the Popup dialogue.

Image

This is a Vector-Popup gadget containing a list of all images used so far. If you wish to wish a new image select the entry "New...". You can then select an IFF file of your choice.

Image-Button attributes

HBorder

Specifies the space left free to the left and right of the image. If you use a border you should not use a value of 0 as the border will then be drawn into the image itself.

VBorder

Specifies the space left free above and below of the image. If you use a border you should not use a value of 0 as the border will then be drawn into the image itself.

Background color

Specifies the color with which the free space is filled. It is recommended to use the default colors offered by the Popup dialogue.

Selected background color

Specifies the color which is used as background color if the Image-Button is selected.

Image

This is a Vector-Popup gadget containing a list of all images used so far. If you wish to wish a new image select the entry "New...". You can then select an IFF file of your choice.

SelImage

Same as Image. The difference is that this image will be shown when the Image-Button is selected.

Note!

Both graphics must use the same dimensions and color depths!

ImageToggle attributes

HBorder

Specifies the space left free to the left and right of the image. If you use a border you should not use a value of 0 as the border will then be drawn into the image itself.

VBorder

Specifies the space left free above and below of the image. If you use a border you should not use a value of 0 as the border will then be drawn into the image itself.

Background color

Specifies the color with which the free space is filled. It is recommended to use the default colors offered by the Popup dialogue.

Selected background color

Specifies the color which is used as background color if the Image-Button is selected.

Image

This is a Vector-Popup gadget containing a list of all images used so far. If you wish to wish a new image select the entry "New...". You can then select an IFF file of your choice.

SelImage

Same as Image. The difference is that this image will be shown when the Image-Button is selected.

Achtung!

Both graphics must use the same dimensions and color depths!

Simple Mode

If this flag is set, the gadget can not be deselected. This is useful if you combine a set of Image-Toggle gadgets using Notifies to form a virtually complex object.

Image-Popup attributes

HBorder

Specifies the space left free to the left and right of the image. If you use a border you should not use a value of 0 as the border will then be drawn into the image itself.

VBorder

Specifies the space left free above and below of the image. If you use a border you should not use a value of 0 as the border will then be drawn into the image itself.

Background color

Da ein Freiraum auch mit einer Farbe geflt werden mu§, empfiehlt es sich diese hier zu spezifizieren. Sie sollten die Farben aus dem Popup-Men verwenden.

Text alignment

Specifies the type of alignment for the Image-Popup menu. You can choose between left-boundary, right-boundary and centered.

Image

This is a Vector-Popup gadget containing a list of all images used so far. If you wish to wish a new image select the entry "New...". You can then select an IFF file of your choice.

3D Look

If this flag is not set the popup menu will appear with a black border on a white background. If the flag is set, the menu will be rendered on a gray background and with a simple border.

Text-Popup attributes

Text alignment

Specifies the type of alignment for the Image-Popup menu. You can choose between left-boundary, right-boundary and centered.

3D Look

If this flag is not set the popup menu will appear with a black border on a white background. If the flag is set, the menu will be rendered on a gray background and with a simple border.

Palette attributes

Colors

This is the number of objects to be shown by this object. Note that this must always be an even number. Specify a 1 if you wish to make the number of colors depend on the current screen.

Offset

The first color to start the palette. Specify 0 to start with the background color.

Selected

A color register which will appear selected when the window is opened.

Vector-Popup attributes

Text alignment

Specifies the type of alignment for the Image-Popup menu. You can choose between left-boundary, right-boundary and centered.

Image

You can choose from three predefined image types: "File", "Drawer" or "Popup".

3D Look

If this flag is not set the popup menu will appear with a black border on a white background. If the flag is set, the menu will be rendered on a gray background and with a simple border.

Hierarchy attributes

ImageWidth

The width in pixels the vector image should be drawn with.

TreeImageWidth

As this object will indent the list in a hierarchic way you must specify the text indention in pixels. If "Lines" was selected as image type this value should be the same as "ImageWidth".

Image type

Choose an Vector-Image to be used to display the hierarchic list.

Doubleclicks

If you want to allow double-clicks on entries be sure to set this flag. A double-click can be recognized by checking the received message's WLISTVIEWA_DoubleClick tag for the value TRUE.

NewFont attributes

Style name

"Style name" is a freely choosable name for the new font. Of course you can also specify the font's original name. It is however often better to use names like "Header" or "Buttontext" as this saves remembering which object you assigned which font!

Proportional

This CheckBox gadget filters the range of fonts which can be selected. If it is checked, you can also select proportional fonts where each character has its own width. Otherwise you can only select monospaced fonts.

Font

Click on this Vector-Button gadget to change the font and/or its size. The font requester will appear which, depending on the Proportional checkbox, will also allow you to select proportional fonts.

Ok

Closes the window and creates the NewFont object. All changes performed are unrecoverably applied.

Cancel

Select this gadget if you wish to abort. All changes will be cancelled and the window will be closed.