

StormWIZARD

COLLABORATORS

	<i>TITLE :</i> StormWIZARD		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY		July 25, 2024	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1	StormWIZARD	1
1.1	StormW.guide	1
1.2	StormWIZARD.guide/Reference	1
1.3	StormWIZARD.guide/ST_Order	2
1.4	StormWIZARD.guide/ST_CRIGHT	3
1.5	StormWIZARD.guide/ST_Lizenz	3
1.6	StormWIZARD.guide/ST_Welcome	5
1.7	StormWIZARD.guide/ST_Philos	5
1.8	StormWIZARD.guide/ST_Maschine	5
1.9	StormWIZARD.guide/ST_Install	6
1.10	StormWIZARD.guide/ST_Tutorial	6
1.11	StormWIZARD.guide/ST_Start	6
1.12	StormWIZARD.guide/ST_Project	7
1.13	StormWIZARD.guide/ST_Project	7
1.14	StormWIZARD.guide/ST_Project	8
1.15	StormWIZARD.guide/ST_Project	10
1.16	StormWIZARD.guide/ST_Project	11
1.17	StormWIZARD.guide/ST_Project	12
1.18	wizard.library/WZ_AllocWindowHandleA	18
1.19	wizard.library/WZ_CloseSurface	20
1.20	wizard.library/WZ_CloseWindow	20
1.21	wizard.library/WZ_CreateWindowObjA	21
1.22	wizard.library/WZ_DrawVImageA	22
1.23	wizard.library/WZ_EasyRequestArgs	30
1.24	wizard.library/WZ_FreeWindowHandle	31
1.25	wizard.library/WZ_GadgetConfig	32
1.26	wizard.library/WZ_GadgetHelp	32
1.27	wizard.library/WZ_GadgetHelpMsg	33
1.28	wizard.library/WZ_GadgetKey	34
1.29	wizard.library/WZ_GetNode	35

1.30	wizard.library/WZ_InitEasyStruct	36
1.31	wizard.library/WZ_ListCount	36
1.32	wizard.library/WZ_LockWindow	37
1.33	wizard.library/WZ_LockWindows	37
1.34	wizard.library/WZ_MenuConfig	38
1.35	wizard.library/WZ_MenuHelp	38
1.36	wizard.library/WZ_NewObjectA	39
1.37	wizard.library/WZ_ObjectID	55
1.38	wizard.library/WZ_OpenSurfaceA	55
1.39	wizard.library/WZ_OpenWindowA()	56
1.40	wizard.library/WZ_SnapShot	57
1.41	wizard.library/WZ_UnlockWindow	58
1.42	wizard.library/WZ_UnlockWindows	58

Chapter 1

StormWIZARD

1.1 StormW.guide

StormWIZARD demo 1.0

Software and Documentation (c) 1996 by HAAGE & PARTNER Computer GmbH

Contents

[License Agreement](#)

[Chapter 1](#) Welcome

[Chapter 2](#) Philosophy of StormWIZARD

[Chapter 3](#) Requirements

[Chapter 4](#) Installation

[Chapter 5](#) Tutorial

[Chapter 6](#) 1st Exercise

[Chapter 7](#) 2nd Exercise

[Chapter 8](#) Paging and Notification

[Chapter 9](#) Layout Technicalities (German)

[Chapter 10](#) Layout Behaviour

[Chapter 11](#) Notify Objects (German)

[Chapter 12](#) Programming (German)

@{ " Reference " Link "Reference" } Library Reference V37.116 (German)

[Copyrights](#)

[Order Form](#)

1.2 StormWIZARD.guide/Reference

Library Reference

[WZ_AllocWindowHandleA\(\)](#)

[WZ_CloseSurface\(\)](#)

[WZ_CloseWindow\(\)](#)

WZ_CreateWindowObjA()
WZ_DrawVImageA()
WZ_EasyRequestArgs()
WZ_FreeWindowHandle()
WZ_GadgetConfig()
WZ_GadgetHelp()
WZ_GadgetHelpMsg()
WZ_GadgetKey()
WZ_GetNode()
WZ_InitEasyStruct()
WZ_ListCount()
WZ_LockWindowA()
WZ_LockWindowsA()
WZ_MenuConfig()
WZ_MenuHelp()
WZ_NewObjectA()
WZ_ObjectID()
WZ_OpenSurfaceA()
WZ_OpenWindowA()
WZ_SnapShot()
WZ_UnlockWindow()
WZ_UnlockWindows()

1.3 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

* I want to order the Cross-Upgrade from my old

Compiler system: _____

at a price of 398 DM

* Yes, please send me StormWIZARD at a price of 98 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.

1.4 StormWIZARD.guide/ST_CRIGHT

Copyrights and trademarks:

Amiga is a registered trademark of VISCORP.

Commodore is a registered trademark of ESCOM AG.

SAS and SAS / C are registered trademarks of the SAS institute.

Amiga, AmigaDOS, Kickstart and Workbench are trademarks of ESCOM Inc.

The designation of products which are not from the HAAGE & PARTNER
COMPUTER Ltd. serves information purposes exclusively and presents no
trademark abuse.

1.5 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

1.6 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.

1.7 StormWIZARD.guide/ST_Philosophy

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.

1.8 StormWIZARD.guide/ST_Maschine

Requirements

You will need the following items to run StormWIZARD:

- any Amiga with hard disk
- Kickstart/Workbench 2.0 (v37) or better
- 2 MB RAM

1.9 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: 100654.3133@compuserve.com

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

1.10 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.

1.11 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 its 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. They 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 ;-).

1.12 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 "BHOOffset" 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 "BVOOffset" to 4. Activate "Redraw" and have a look at the result - now everything is fine.

1.13 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, "BHOOffset" to 6 and "BVOOffset" 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.

1.14 StormWIZARD.guide/ST_Project

Layouttechniken

Um die Position von Objekten innerhalb eines Fenster automatisch bestimmen zu lassen, werden diese in Gruppen-Gadgets untergebracht. Wir unterscheiden horizontale und vertikale Gruppen.

1. Merksatz:

Ein horizontales Gruppen-Gadget ordnet seine Mitglieder nebeneinander an, während das vertikale Gruppen-Gadget die Mitglieder untereinander anordnet.

2. Merksatz:

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

3. Merksatz:

Die Mindestbreite einer horizontalen Gruppe für die übergeordnete Gruppe setzt sich aus der Summe aller minimalen Mitgliederbreiten und den freien Plätzen zwischen diesen zusammen. Außerdem muß der Wert für "HBorder" doppelt addiert werden. Die Mindesthöhe ergibt sich aus dem doppelten Wert für VBorder und der größten Mindesthöhe eines Mitglieds.

4. Merksatz:

Die Mindestbreite einer vertikalen Gruppe für die übergeordnete Gruppe errechnet sich aus dem doppelten Wert für "HBorder" und der größten Mindestbreite eines Mitglieds. Dagegen muß für die Mindesthöhe einer Gruppe die Summe aller Einzelmindesthöhen der Mitglieder mit den dazwischenliegenden Freiräumen addiert werden. Der Wert für "VBorder" wird wieder doppelt hinzugerechnet.

5. Merksatz:

Innerhalb einer horizontalen Gruppe bekommt ein Mitglied immer die volle Höhe des Gruppengadgets zugeteilt. Dabei wird natürlich "VBorder" oben und unten freigelassen. Ein vertikales Gruppengadget übergibt allen Mitgliedern die selbe Breite, unter Berücksichtigung von "HBorder" an der linken und rechten Seite.

6. Merksatz:

Ein horizontale Gruppen-Gadget errechnet die Breite eines Mitgliedes anhand dessen Priorität und seiner minimalen Breite. Dabei beachtet es auch die minimalen Breiten aller anderen Mitglieder. Der "HBorder"-Wert wird dabei links und rechts freigelassen. Umgekehrt gilt für vertikale Gruppen das der "VBorder"-wert oben und unten freigelassen wird und mit dem verbleibenden Wert wird die Höhe für jedes Mitglied errechnet. Dabei beachtet dieses die minimale Höhe und Priorität des Einzelobjektes wie auch die Gesamtsumme aller Mitglieder. Den entscheidenden Einfluß auf die Verteilung des Platzes unter den Mitgliedern einer Gruppe, übt also die Priorität der Mitglieder untereinander aus!

7. Merksatz:

Besitzt ein Mitglied einer Gruppe eine Priorität von Null, dann bedeutet es, das innerhalb einer horizontalen Gruppe seine Breite immer auf den minimalen Betrag gesetzt wird! Bei einer vertikalen Gruppe trifft dies dann auf die Höhe des Mitglieds zu.

8. Merksatz:

Besitzt ein horizontales Gruppen-Gadget das Flag "Equalsize", dann bedeutet es, das alle Mitglieder die selbe Mindestbreite verordnet bekommen. Dabei fragt das Gruppengadget alle Mitglieder nach der Mindestbreite und rechnet mit der größten Mindestbreite für alle Mitglieder weiter. Das vertikale Gruppen-Gadget funktioniert entsprechend für die Mindesthöhe.

9. Merksatz:

Beträgt die Gesamtpriorität aller Mitglieder einer Gruppe gleich Null, darf der Wert für "Var. Leerraum" nur mit 0 oder 100 Prozent initialisiert werden. Geben Sie hier trotzdem einen davon abweichenden Wert ein, müssen Sie mit einer nicht definierten Positionsbestimmung rechnen.

In zukünftigen Versionen der Library kann sich hier etwas ändern!

10. Merksatz:

Ein Objekt ordnet sich immer seiner übergeordneten Gruppen ein. Dies geschieht unter Berücksichtigung der beim Mitglied festgelegten Seite. Wurde bei einem Objekt die Seitennummer 1 eingegeben, dann ist es nur sichtbar, wenn die Seite 1 beim Gruppengadget die aktuelle Seite ist.

11. Merksatz:

Der "Dockmodus" eines Gruppen-Gadgets berücksichtigt nur die minimale Breite und Höhe eines Mitglieds. "Leerraum" und "Var. Leerraum" werden ebenfalls nicht berücksichtigt. Für diesen Modus muß das Gruppengadget mit einem Link zu einem Proportional-Gadget versehen werden, welches rechts oder unter dem Gruppengadget angeordnet ist!

1.15 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.

1.16 StormWIZARD.guide/ST_Project

Die Notifyobjekte

Da jede Klasse ihre besonderen Eigenschaften hat und es sinnvoll ist, diese zu kombinieren, können Objekte miteinander kommunizieren.

Dies geschieht ohne Zutun des Programmierers. Man kann zum Beispiel einen Slidergadget mit einem Integergadget zusammenarbeiten lassen.

Dazu muß jedoch ein Verbindungs-Objekt angelegt werden. Für solche Arbeiten wurde der Notify-Editor entworfen. Dieses Verbindungs-Objekt kann, wenn eine Nachricht vom Source-Objekt abgeschickt wird, "mappen".

Dieses Mappen bedeutet, das eine Informationskennung geändert wird.

So lässt sich zum Beispiel die Informationskennung WSLIDERA_Level, mit welcher der Stand eines Sliders gekennzeichnet wird, in die

Informationskennung WINTEGERA_Long "mappen".

Diese neue Informationskennung wird von unserem Zielobjekt, einem Integer-Gadget, verstanden. Es lassen sich prinzipiell fast alle Klassen miteinander verbinden.

Ändert der Benutzer aber nicht den Sliderstand, sondern den Wert des Integergadgets, wird ersterer nicht benachrichtigt. Zu diesem Zweck sollten Sie ebenfalls ein Verbindungsobjekt erzeugen, welches die Informationskennung von WINTEGERA_Long nach WSLIDERA_Level konvertiert.

Damit sind beide Objekte in beiden Richtungen miteinander verknüpft. Es läßt aber nicht nur die Informationskennung ändern, sondern auch die Information selbst.

1.17 StormWIZARD.guide/ST_Project

Nutzung der Wizarddateien und Programmierung der Oberflächen!

Dies dürfte wohl das interessanteste Kapitel sein, deshalb finden Sie hier auch Beispiele wie Funktionen in der Sprache ANSI-C angesprochen werden.

Um auch die Funktionen der Library nutzen zu können, muß diese natürlich auch geöffnet werden, dies geschieht wie auf dem Amiga üblich:

```
struct Library *WizardBase; // Basisadresse definieren
```

```
.  
.   
.
```

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

```
{
```

```
// Library geöffnet
```

```
}
```

```
else
```

```
// Fehlerabfang!
```

Am Ende eines Programmes sollten Sie die Library wieder freigeben, damit der Speicher der dafür belegt wurde, freigegeben werden kann.

Das könnte wie folgt aussehen:

```
.  
.   
.
```

```
if( WizardBase)
```

```
CloseLibrary( WizardBase);
```

```
// ab hier keine Funktion mehr nutzbar!!!
```


Um jetzt eine Oberflächenbeschreibung verfügbar machen zu können, sollten Sie die Funktion WZ_OpenSurface() benutzen. Ein Beispiel:

```
APTR MySurface;
```

```
.  
.   
.
```

```
if( (MySurface = WZ_OpenSurface("manager.wizard",0L,TAG_DONE)))
```

```
{
```

```
// Datei erfolgreich geladen
```

```
}
```

```
else
```

```
// Fehlerabfang!
```

Zurückgegeben wird die Speicheradresse, an die die Wizard-Datei geladen wurde. Im Fehlerfall erhalten Sie eine Null als Ergebnis.

Wenn Sie die Lokalisierungseigenschaften von StromWizard nutzen möchten, muß die Adresse des entsprechenden Lokale-Catalogs beim Öffnen der Oberflächendatei mitangegeben werden.

Zuvor muß der Catalog allerdings mit den Funktionen der Locale.Library ebenfalls geladen werden.

Beachten Sie, daß Sie hierbei auch die locale.library geöffnet haben müssen.

Beispiel:

```
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);
```

```
}
```

Die Abfrage, ob der Katalog geladen werden konnte ist nicht unbedingt erforderlich! Sie können also stur programmieren.

Der im obigen Beispiel übergeben Null-Parameter ist ein Zeiger auf eine

Adresse innerhalb des Speichers. Falls nämlich die ".wizard"-Datei bereits im Speicher ist (durch Binärinclude), dann sollten Sie diese Adresse angeben und den Namensstring auf Null setzen.

Beispiel:

```
MySurface=WZ_OpenSurface(OL,SurfaceAddress,TAG_DONE);
```

Nun wollen wir aber ein Fenster erzeugen und dazu müssen wir erst einmal ein sogenanntes WindowHandle beantragen. Dazu machen wir einen Funktionsaufruf wie im Folgenden beschrieben:

```
struct WizardWindowHandle *MyWinHandle;
```

```
MyWinHandle=WZ_AllocWindowHandle(MyScreen,  
sizeof(MyWinExtension),  
MySurface,  
TAG_DONE);
```

MyScreen ist ein Zeiger auf den zu verwendenden Screen, auf dem unser Fenster später erscheinen soll.

Gleichzeitig können Sie die Größe einer automatisch bereitzustellenden privaten Struktur angeben. Dies ist nützlich, da man häufig eigene Daten zu einem Fenster verwalten muß. Die Adresse dieser bereitgestellten Struktur finden Sie in der WizardWindowHandle-Struktur eingetragen.

Möchten Sie von dieser Möglichkeit keinen Gebrauch machen, geben Sie hier eine Null an.

MySurface ist der vom vorhergehenden WZ_OpenSurface zurückgegebene Handle.

Zu guter Letzt können noch Tags übergeben werden. Im Moment sind für kann nur das Tag WWH_StackSize angegeben werden. Dieses Tag beschreibt die Stackgröße des Fensterspezifischen Stacks, welcher beim Layoutvorgang benutzt werden soll. Dieser wird gleichzeitig beim Funktionsaufruf allokiert.

Wenn Sie den WindowHandle nicht mehr benötigen sollten Sie ihn wieder freigeben, dabei werden alle Objekte, die in dem WindowHandle eingetragen sind automatisch wieder freigegeben. Dies betrifft auch ein eventuell geöffnete Fenster!

```
WZ_FreeWindowHandle( MyWinHandle);
```

Ein WindowHandle alleine ist nicht sinnvoll, deshalb gibt es eine Funktion zum Erzeugen aller zu einem Fenster gehörenden Objekte. Dies betrifft Gadgets, Menüs, Notify-Verbindungsobjekte und anderes.

Beispiel:

```
#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);

```

Der Parameter MyWinHandle benötigt keine Erklärung mehr, dafür aber MY_WINDOW_ID. Hierbei handelt es sich um die Identifizierungsnummer, welche von StormWizard beim Speichern vergeben wird.

In der automatisch mitgespeicherten Include-Datei finden Sie diese ID unter dem im StormWizard im Fensterattribut-Einsteller eingegeben ObjektNamen.

Die folgendende Tagliste sollte mindestens das Tag WWH_GadgetArray enthalten, mit dem Sie die Adresse ihrer Gadget-Liste übergeben. Denn Sie wollen schließlich auch auf die erzeugten Gadgets zugreifen können.

Dabei sollte das Array alle Gadgets aufnehmen können. Um also sicher zu stellen, das das GadgetArray nur akzeptiert wird, wenn es groß genug ist, sollten Sie außerdem die Größe in Bytes beim Tag WWH_GadgetArraySize angeben.

Konnten alle Objekte richtig zurückgegeben werden, dann erhalten Sie die Adresse einer bereits initialisierten NewWindow-Struktur zurück.

Diese Struktur dürfen Sie selbst ändern und brauchen Sie auch für den Aufruf in der folgenden Funktion:

```

struct Window *MyWindow;
MyWindow=WZ_OpenWindow( MyWinHandle,
MyNewWindow,
WA_AutoAdjust ,TRUE,
TAG_DONE);

```

Die Parameter MyWinHandle und MyNewWindow kennen Sie bereits von den vorhergehenden Funktionen. Hinzu kommt dann eine Tagliste, die alle Tags zulässt, welche bei der Funktion "OpenWindowTagList()" zulässig sind.

Es empfiehlt sich dabei, das Tag WA_AutoAdjust auf TRUE zu setzen, um das Fenster notfalls anpassen zu lassen.

Um das Fenster zu schließen existiert analog die Funktion:

```
WZ_CloseWindow( MyWinhandle);
```

Dabei können Sie ein Fenster beliebig oft öffnen und wieder schließen!

Da ein Fenster nur Sinn macht, wenn man auch Nachrichten empfängt, sollten Sie mittels Wait() oder WaitPort() auch auf Nachrichten warten.

Holen Sie die Nachricht wie gewohnt mittels GetMsg() vom Userport des

Fensters ab. Möchten Sie nun die Tastaturunterstützung für Wizard-Klassen nutzen, schauen Sie sich folgenden Auszug an.

```
switch(msg->Class)
{
case IDCMP_VANILLAKEY:
WZ_GadgetKey( MyWinHandle, msg->Code, msg->Qualifier, TAG_DONE);
break;
.
.
.
}
```

Die Besonderheit ist die, daß, falls ein Gadget für diesen Tastendruck zuständig war, dieses veranlasst wird, selbst eine Nachricht vom Type IDCMP_IDCMPUPDATE abzuschicken. Dabei werden die Notify-Objekte ebenfalls berücksichtigt.

Handelte es sich um ein Integer- oder String-Gadget, dann wird dieses aktiviert und es sendet keine Nachricht aus. Die Funktion gibt außerdem zurück, ob überhaupt ein Gadget zuständig gewesen ist.

In unserem Beispiel haben wir das aber nicht berücksichtigt.

Ab der Betriebssystemversion V39 kann der Amiga IDCMP_GADGETHELP`'s an den Programmierer senden. Häufig nutzt man das, um einen Hilfe-Text für den Anwender darzustellen. Dazu können Sie mit einer Funktion einen Hilfe-Text für jedes Objekt im StormWizard eingeben. Anfordern können Sie ihn bei einem Gadget wie folgt.

STRPTR Help;

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

Unter Umständen können Sie auch eine Nulladresse bekommen, beachten Sie dies, in dem Sie das Ergebnis abfragen. Auch für Menüs gibt es eine solche Funktion, falls IDCMP_MENUHELP - Nachrichten ankommen.

STRPTR Help;

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

Leider kann erst ab Version 39 das Betriebssystem eine solche Funktion übernehmen. Häufig soll die Entwicklung aber schon ab V37 (OS 2.0/2.1) laufen, deshalb besitzt die Library auch eine Funktion, mit der die Gadget-Help Nachricht simuliert werden kann. Dazu müssen Sie die Mausbewegung beobachten.

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 ist das Fenster, bei dem die Nachricht einer Mausbewegung angekommen ist.

HelpWinHandle enthält den Zeiger auf ein WizardWindowHandle, wenn eine Gadget-Help Nachricht vorliegt.

Die Variable HelpIAddress enthält dabei den Wert, wie er bei IntuiMessage->IAddress zu finden ist.

Die Mausposition wird natürlich auch benötigt und noch Flags, welche das Verhalten der Funktion bestimmen.

Wird das Flag WGHF_IgnoreOS gesetzt, wird auch ab OS V39 und höher eine GadgetHelpfunktion simuliert, sonst nur unter V37 (OS 2.0/2.1).

Dagegen besitzt das Flag WGHF_FullContol noch keine Bedeutung.

Häufig ist es erforderlich ein geöffnetes Fenster gegen Eingaben von Seiten des Benutzers zu sperren, auch dafür besitzt die Library eine Funktion, der Sie das WizardWindowHandle übergeben müssen. Sie dürfen diese Funktion mehrmals hintereinander aufrufen:

```
WZ_LockWindow( MyWinHandle); // Fenster ist jetzt gesperrt
```

Um es wieder für Eingaben zuzulassen, existiert auch eine Funktion, welche den selben Parameter verlangt.

```
WZ_UnlockWindow(MyWinHandle); // Fenster wieder bereit für Eingaben
```

Dabei wird das Fenster nur wieder für Eingaben zulässig, wenn diese Funktion genauso oft aufgerufen wird, wie die Funktion WZ_LockWindow() für dieses Fenster.

Wenn man mit mehreren geöffneten Fenstern arbeitet, dann können Sie schlagartig mit einer Funktion alle geöffneten Fenster gegen Eingaben sperren.

Als Parameter dient die von WZ_OpenSurface() gelieferte Adresse.

Achten Sie auf das angefügte "s" !

```
WZ_LockWindows( MySurface);
```

Das Gegenteil bewirkt die Funktion:

```
WZ_UnlockWindows( MySurace);
```

1.18 wizard.library/WZ_AllocWindowHandleA

NAME

WZ_AllocWindowHandleA -- WizardWindowHandle anlegen

WZ_AllocWindowHandle -- variable Parameterübergabe für

Hochsprachen

SYNOPSIS

```
winhandle = WZ_AllocWindowHandleA(
screen, user_sizeof, surface, tags )
```

D0 D0 D1 A0 A1

```
struct WizardWindowHandle *WZ_AllocWindowHandleA(
struct Screen *, ULONG, APTR, struct TagItem *);
```

```
winhandle = WZ_AllocWindowHandle(
screen, user_sizeof, surface, firstTag, ... )
```

```
struct WizardWindowHandle *WZ_AllocWindowHandle(
struct Screen *, ULONG, APTR, Tag, ...);
```

FUNCTION

Belegt den Speicher für einen WizardWindowHandle. Gleichzeitig wird dieser natürlich initialisiert.

WARNING

Diese Struktur ist intern noch mit zusätzlichen Feldern definiert.

INPUTS

screen - ein Zeiger auf die Screen-Struktur, auf dem das Fenster später erscheinen soll

user_sizeof - die Größe in Bytes die eine private Struktur haben soll, mit der man eigene Daten zum Fenster verwalten will, siehe WZ_WindowUserStruct().

surface - der Returnwert von WZ_OpenSurface()

tags

WWH_StackSize, ULONG

Größe des Stacks, der für den Layoutvorgang bereitgestellt werden soll. (Vorgabe 8192).

RESULT

winhandle - Zeiger auf eine WizardWindowHandle-Struktur oder Null im Fehlerfall

Window - ein Zeiger auf die Fensterstruktur von Intuition, falls das Fenster geöffnet ist, ansonsten eine Null.

MenuStrip - Zeiger auf den MenuStrip dieses Fensters oder Null, wenn kein Menu existiert.

DrawInfo - wird beim Anlegen der Struktur von der Library ausgefüllt.

VisualInfo - wird beim Anlegen dieser Struktur von der Library ausgefüllt.

ScreenTitle - in diesem Feld sollte der Screenshot des Fensters eingetragen sein, welcher beim Öffnen des Fensters mittels WZ_OpenWindowTags() benutzt wird. Dieses Feld wird beim Anlegen dieser Struktur mit -1 ausgefüllt. Bei Aufruf von WZ_CreateWindowObj() setzt diesen Wert auf einen sinnvollen Wert.

Objects - Eine Minlist-Struktur, in der BOOPSI-Objekte verkettet sind, welche beim Freigeben dieser Struktur mittels intuition.library/DisposeObject() ebenfalls entfernt werden sollen. Das Eintragen eigener Objekte ist möglich.

RootGadget - ein Zeiger auf das Gruppengadget, welches innerhalb des Fensters seine Mitglieder platziert und für diese auch den Layoutvorgang initialisiert.

Ein RootGadget existiert IMMER, wenn alle Objecte die zu einem Fenster gehören, angelegt wurden, Gleichzeitig ist dieses Gadget der Anfang der gesamten Gadgetliste !

RootTopGadget

RootLeftGadget

RootBottomGadget

RootRightGadget

Diese Felder werden erst in späteren Versionen dieser Library eine Bedeutung bekommen. Im Moment sind diese Felder auf Null gesetzt.

UserStruct - ein Zeiger auf eine private Struktur, welche mit Nullwerten vorinitialisiert ist. War der Funktionsaufruf mit dem Parameter user_sizeof = Null, dann ist dieser Zeiger mit Null initialisiert und darf nicht als solcher benutzt werden.

Das verwenden dieses Feldes zum Selbsteintragen einer Struktur oder eines anderen Wertes ist in einem solchen Fall aber gestattet !

SEE ALSO

WZ_CreateWindowObjA(), WZ_OpenWindowA(), WZ_CloseWindow(),
WZ_FreeWindowHandle()

1.19 wizard.library/WZ_CloseSurface

NAME

WZ_CloseSurface -- Oberflächenbeschreibung abmelden

SYNOPSIS

WZ_CloseSurface(surface)

A0

VOID WZ_CloseSurface(APTR);

FUNCTION

Diese Funktion meldet eine Oberflächenbeschreibung ab und gibt das belegte Ram wieder frei.

WARNING

Sind noch WindowHandles oder geöffnete Fenster vorhanden, werden diese geschlossen und entfernt. Alle Ressourcen sind nicht mehr ansprechbar.

INPUTS

surface - der Returnwert von WZ_OpenSurface()

SEE ALSO

WZ_OpenSurfaceA(),WZ_SnapShot()

1.20 wizard.library/WZ_CloseWindow

NAME

WZ_CloseWindow -- Fenster schliessen

SYNOPSIS

WZ_CloseWindow(winhandle)

A0

VOID WZ_CloseWindow(struct WizardWindowHandle *);

FUNCTION

schliesst ein Fenster ähnlich dem Intuitionsaufruf.

WARNING

Das Benutzen der Intuition-Funktion ist nicht erlaubt.

INPUTS

winhandle - WizardWindowHandle von WZ_AllocWindowHandle()

SEE ALSO

WZ_OpenWindowA()

1.21 wizard.library/WZ_CreateWindowObjA

NAME

WZ_CreateWindowObjA -- Fenster-Objecte anlegen

WZ_CreateWindowObj -- variable Parameterübergabe für Hochsprachen-programmierer

SYNOPSIS

```
newwin = WZ_CreateWindowObjA(winhandle, id, tags)
```

```
D0 A0 D0 A1
```

```
struct NewWindow *WZ_CreateWindowObjA
```

```
(ULONG, struct WizardWindowHandle *,struct TagItem *);
```

```
newwin = WZ_CreateWindowObj(winhandle, id, firstTag, ... )
```

```
struct NewWindow *WZ_CreateWindowObj
```

```
(ULONG, struct WizardWindowHandle *,Tag, ... );
```

FUNCTION

Mit diesem Funktionsaufruf werden alle für das Fenster wichtigen Objecte (Gadgets, Menu, Notifyobjecte) angelegt.

WARNING

Die Beschreibung der Objecte kann NICHT kontrolliert werden.

INPUTS

id - Nummer des anzulegenden Windows

winhandle - Zeiger auf einen von WZ_AllowWindowHandle()

besorgten WizardWindowHandle

tags - TagItems, um z.B. das GadgetArray zu übergeben

WWH_GadgetArray - ein Zeiger auf ein Array, in dem die erzeugten Gadgets nach der GadgetID abgelegt werden.

Diese Tag wird immer verlangt !

WWH_GadgetArraySize - Größe des Arrays in Bytes zur Sicherheit.

dieses Tag darf wegelassen werden.

WWH_PreviousGadget - Gadget, hinter dem die zu erzeugenden Gadgets eingehangen werden sollen

WWH_StringHook - Zeiger auf eine StringHook-Struktur, welche an alle Stringgadgets in dem Fenster übergeben

werden soll. Genauere Information finden Sie

bei der Beschreibung der "strgclass" !

RESULT

newwin - Zeiger auf eine initialisierte NewWindow-Struktur

oder Null im Fehlerfall. Diese ist wie folgt

vorinitialisiert:

LeftEdge,
TopEdge,
Width,
Height - Werte die in der Oberflächenbeschreibung abgespeichert sind. Diese lassen sich wieder mit **WZ_SnapShot()** fixieren.
DetailPen,
BlockPen - mit ~0 vorbelegt
IDCMPFlags - stammen aus der Oberflächenbeschreibung
Flags - ebenfalls aus der Oberflächenbeschreibung
FirstGadget - RootGadget des Fensters
CheckMark - Null
Title - String der in StormWizard festgelegt wurde.
Screen - Screenstruktur, welche bei WZ_AllocWindowHandle() übergeben wurde.
BitMap - Null
MinWidth,
MinHeight - minimale Dimension, die das Fenster haben darf, diese Werte werden jedesmal aufs Neue berechnet.
MaxWidth,
MaxHeight - diese Werte werden mit ~0 initialisiert.
Type - mit dem Wert von CUSTONSCREEN.
SEE ALSO
WZ_AllocWindowHandle(), **WZ_FreeWindowHandle()**, WZ_OpenWindow(),
WZ_CloseWindow()

1.22 wizard.library/WZ_DrawVImageA

NAME

WZ_DrawVImageA -- Vektorgrafik zeichnen

WZ_DrawVImage -- für Hochsprachenprogrammierer mit variabler Parameterübergabe

SYNOPSIS

success = WZ_DrawVImageA

(vimage, x, y, w, h, type, rp, drinfo, tags)

D0 A0 D0 D1 D2 D3 D4 D5 D6 A1

BOOL WZ_DrawVImageA(

struct WizardVImage *,

```

WORD, WORD, WORD, WORD, UWORD, struct RastPort *,
struct DrawInfo *, struct TagItem *);
success = WZ_DrawVImage
(vimage, x, y, w, h, type, rp, drinfo, firstTag, ... )
BOOL WZ_DrawVImage(
struct WizardVImage *,
WORD, WORD, WORD, WORD, UWORD, struct RastPort *,
struct DrawInfo *, Tag, ... );
FUNCTION

```

Diese Funktion zeichnet eine Vektorgrafik mit den angegebenen Parametern. Es sind damit Turtle-Graphics realisierbar.

WARNING

Teilweise sind in der Definition einer Vektorgrafik Verweise auf andere Vektorgrafiken vorhanden. Dabei ruft sich diese Funktion selbst auf. Der Stack wird dabei gnadenlos genutzt !

INPUTS

vimage - Zeiger die WizardVImage-Struktur die wie folgt aufgebaut ist :

flags

WVIF_MinWidth - wenn die zum Zeichnen übergebene Breite kleiner ist als die in der WizardVImage-Struktur übergebene, dann wird im Normalfall nicht gezeichnet. Dieses Flag zwingt in einem solchen Fall die übergeben Breite auf diesen Mindestwert, dabei wird die X-Position zur Hälfte der Differenz nach links korrigiert.

WVIF_MinHeight - wenn die übergebene Höhe mindestens den Wert aus WizardVImage->MinHeight haben soll. Eine Korrektur nach oben findet in einem solchen Fall ebenfalls statt. Sollte dieses Flag nicht gesetzt sein und die Höhe wird unterschritten, dann wird die Funktion WZ_DrawVImage() nicht durchgeführt.

WVIF_AreaInit - dieses Flag besitzt zur Zeit noch keine Bedeutung.

WVIF_Recursion - zeigt an, das diese Vektorgrafik sich selbst aufruft und dabei der Wert, der in VImage->Counter steht, bei jedem Aufruf dieser Grafik um eins erniedrigt wird. Wird der Wert Null erreicht, dann wird die Rekursion abgebrochen. Dieses Flag darf nur verwendet werden, wenn die Vektorgrafik nur einem einzigen

Task zur Verfügung steht.

Counter - sollte das Flag WVIF_Recursion gesetzt sein, dann wird hiermit die maximale Tiefe bei einer Rekursion angegeben.

Möchten Sie das die Grafik nach dem Funktionsaufruf von WZ_DrawVImage() vom Task aus noch ein einziges mal aufgerufen wird, dann geben Sie hier eine 2 an.

MinWidth - Geben Sie hier die Mindestbreite an, mit der ein Funktionsaufruf von WZ_DrawVImage() erfolgen muß.
siehe Flag WVIF_MinWidth

MinHeight - Geben Sie die Mindesthöhe für einen Aufruf von WZ_DrawVImage() an. siehe Flag WVIF_MinHeight

RelCoords - Möchten Sie eigene Bezugspunkte angeben, dann geben Sie hier einen Zeiger auf eine UWORD-Tabelle an.
dessen 1.Word dem relativen X und dessen 2.Word dem relativen Y entspricht. Terminieren Sie diese Tabelle mit zwei aufeinanderfolgenden Nullen.

Der 1.Eintrag in dieser Tabelle entspricht dem Bezugspunkt mit der Nummer 4 ! Die einzutragenden X und Y Werte müssen sich im Bereich von 0 bis 65535 befinden. Wobei eine Null links bzw. oben entspricht. Vordefinierte Bezugspunkte sind : 0 - links und oben, 1 - rechts und oben, 2 - rechts und unten sowie 3 - links und unten.

Wenn Sie diesen Zeiger mit Null ausfüllen, dann dürfen Sie nur die Bezugspunkte 0 bis 3 nutzen !

Images - hier können beliebig viele aufeinander folgende Zeiger eingetragen werden. Der übergebene type bestimmt dann, welcher Zeiger und damit welche Zeichendefinitionen benutzt wird. Wurde type mit 0 angegeben, dann wird direkt der Zeiger aus WizardVImage->Images genommen, ansonsten einer der darauf folgenden. Gleichzeitig ist damit bedingt, das die Länge der Struktur NICHT festgelegt werden kann.

x - linke Position der zu zeichnenden Grafik

y - obere Position

w - Breite

h - Höhe

type - Type der zu zeichnenden Grafik

rp - Zeiger auf den zu verwendenden RastPort

drinfo - Zeiger auf eine DrawInfo-Struktur, wenn DrawInfo-Farben

benutzt werden sollen oder Textfunktionen im Image vorkommen.

tags - folgende Tags sind definiert:

WVIA_Text (V1), STRPTR

Adresse eines mit NullByte oder Return abgeschlossenen Strings, für Text-Kommandos.

WVIA_TextFont (V1), struct TextFont *

TextFont-Struktur, der bei Textfunktionen genommen werden soll. Vorgabe ist der DrawInfo-Zeichensatz.

WVIA_TextPlace (V1), ULONG

beschreibt die Textausgabe mittels der Konstanten:

WZRDPLACE_LEFT, WZRDPLACE_CENTER, WZRDPLACE_RIGHT für links, mittig, rechts. (WZRDPLACE_CENTER)

WVIA_TextPen (V1), UWORD

dieses Tag hat zur Zeit noch keine Funktion, setzen Sie die Farbe vorher mit einem entsprechendem Kommando vorher.

WVIA_TextStyles (V1), ULONG

dieses Tag hat zur Zeit noch keine Funktion, setzen Sie den Style mit einem entsprechendem Kommando vorher.

WVIA_TextHighlights (V1), ULONG

dieses Tag hat zur Zeit noch keine Funktion.

WVIA_TextImages (V1), BOOL

TRUE, um das Zeichnen von Images mit der Dimension von einer TextAusgabe zuzulassen. FALSE, wenn diese ImageArt ausgeschlossen werden soll. (TRUE)

WVIA_TagImage (V1), struct WizardVImage *

Ermöglicht die Angabe einer variablen Vektorimage-definition.

WVIA_TagImageCode (V1), UWORD

Beschreibt den Type, mit der das in WVIA_TagImage angegebene VImage gezeichnet werden soll. (0)

WVIA_ImageCode (V1), UWORD

Falls in einem festen Verweis auf ein anderes VImage der zu zeichnende Type mit -1 festgelegt wurde, dann kann hier der richtige Type für diese VImage angegeben werden. Fehlt trotzdem diese Tag, wird der Imageverweis nicht bearbeitet. (-1)

WVIA_Color0-7 (V1), UWORD

Diese Tags enthalten Farben, die mittels des Kommandos

WVICMD_TAGCOLOR angesprochen werden können. Es können auch DrawInfo-Farben gewählt werden.

WVIA_TPoint0-7 (V1), struct TPoint

Hier können von außen berechnete Positionen in die VImage-Definition eingebracht werden. ($X < 16 + Y$)

WVIA_AreaPtrn (V1), APTR

Ein Zeiger auf ein zu benutzendes Raster für alle folgenden flächenfüllenden Kommandos (WVICMD_RECTFILL).

Die Höhe wird direkt beim Kommandoaufruf WVICMD_SETAFPT angegeben.

WVIA_TmpRas (V1), struct TmpRas *

Hier muß eine initialisierte TmpRas-Struktur angegeben werden, um mittels WVICMD_AREAINIT im RastPort fixiert zu werden.

WVIA_BitMapWidth (V1), UWORD

Die Breite der in speziellen Tags angegebenen Bitmap. (0)

WVIA_BitMapHeight (V1), UWORD

Die Höhe der in speziellen Tags angegebenen Bitmap. (0)

WVIA_BitMap0-7 (V1), struct BitMap *

Bitmaps für DrawVImage-Kommandos.

WVIA_PureText (V1), BOOL

TRUE, wenn ein Unterstrich das Unterstreichen des folgenden Buchstabens kennzeichnen soll, sonst FALSE.

(Vorgabe TRUE).

Kommandos für VImage-Definitionen:

Alle Kommandos sind vom Type LONG !

Auch der Platzverbrauch aller folgenden Parameter ist identisch mit der Größe LONG !!!

WVICMD_END :

Signalisiert das Ende einer Definition.

WVICMD_COLOR : Pen;

Setzt den APen des Rastports auf den folgenden Pen, dabei sind DrawInfo-Farben zugelassen.

Bereich 0 - 255 oder WZRD_... -Farben

WVICMD_COLOR2 : Pen(;

Setzt den BPen des Rastports auf den folgenden Pen, dabei sind DrawInfo-Farben zugelassen. Wird nur selten benötigt.

WVICMD_MOVE : BPkt, XOffset, YOffset;

Bewegt den Grafikkursor auf den Bezugspunkt, um die in XOffset und YOffset angegebenen Werte zu verschieben.

WVICMD_DRAW : BPkt, XOffset, YOffset, Mask;

Zieht eine Linie von der aktuellen Grafikkursorposition zum angegebenen Bezugspunkt, dabei werden XOffset und YOffset zum Bezugspunkt dazu addiert. Mit Maske wird das Linienmuster definiert. Der eingestellte APen wird benutzt.

WVICMD_RECTFILL : BPkt, XOffset, YOffset;

Zeichnet von der aktuellen Position des Cursors zum angegebenen Bezugspunkt eine ausgefüllte Fläche mit dem APen. Dabei muß der Cursor bereits auf der linken oberen Ecke des zu zeichnenden Rechtecks stehen. Der durch den Bezugspunkt definierte Grafikpunkt muß die rechte untere Ecke kennzeichnen.

WVICMD_WRITEPIXEL : BPkt, XOffset, YOffset;

Zeichnet an den um XOffset und YOffset verschobenen Bezugspunkt einen farbigen Punkt, in der APen-Farbe.

WVICMD_IMAGE : BPkt, XOffset, YOffset, Type, VImage;

Führt recursiv einen VImage-Aufruf durch. Dabei muß die linke obere Ecke vorher mit dem Kommando WVICMD_MOVE festgelegt worden sein. Die rechte untere Ecke bestimmt sich aus dem Bezugspunkt und dem X- sowie YOffset. Type gibt für den Aufruf den zu benutzenden Type an, der eine -1 enthalten muß, falls der eigentliche Type dem Tag WVIA_ImageCode entnommen werden soll. VImage ist ein Zeiger auf die Vektorgrafik, die hier gezeichnet werden soll.

WVICMD_TEXT :

Geben Sie für dieses Kommando immer mindestens das Tag WVIA_Text an ! Mit dieser Funktion wird der angegebene Text an die aktuelle Position gezeichnet. Setzen Sie vorher also Farbe, Schnitt und Position.

WVICMD_SETDRMD :

Hiermit wird der DrawMode des RastPort umgestellt. Bei Beginn des DrawImageaufrufes ist dieser immer RP_JAM1 !

WVICMD_TEXTIMAGE : HBorder, VBorder, Type, VImage;

Wenn ein Text ausgegeben wird, dann ist dieser an einer

Position und hat eine bestimmte Ausdehnung. Hiermit wird diese Position für einen Imageaufruf genommen. Dabei wird aber nach links und rechts die Ausdehnung um den Wert HBorder vergrößert.

Dies gilt auch für die vertikale Ausdehnung durch VBorder. Der Imageaufruf erfolgt mit dem angegebenen Type und dem angegebenen Vektorimage.

Diese Art von Images kann mit dem Tag WVIA_TextImages für das aktuelle Vektorimage abgeschaltet werden.

WVICMD_TEXTMOVE : FontSizeFak, BaseFak, LengthFak;

Diese Funktion verschiebt den Cursor von der aktuellen Position um einen bestimmten Wert. Dabei wird die Länge der möglichen Textausgabe durch LengthFak dividiert und zur aktuellen X-Position hinzuaddiert.

Den Wert TextFont->Baseline dividiert er durch BasekFak und verschiebt den Cursor in der Y-Richtung.

Die Zeichensatzhöhe dividiert er durch FontSizeFak und verschiebt den Cursor ebenfalls in der Y-Richtung.

Bei allen Parametern dieses Kommandos kann eine Null angegeben werden, um ihn wirkungslos zu machen.

WVICMD_TAGCOLOR : Col;

Col enthält die Nummer des WVIA_Color - Tags, dessen Farbe gesetzt werden soll. Um die Farbe des Tags

WVIA_Color1 setzen zu können muß dieser Parameter eine Eins enthalten.

WVICMD_TEXTPLACE: BPKT, XOffset, YOffset;

Um komfortabel TextPositionen angeben zu können, wurde diese Funktion geschrieben. Hier wird der TextCursor so plziert, das der Text vertikal zentriert in einem definierten Rechteck ausgegeben wird.

Mit dem Tag WVIA_TextPlace kann der Text links, mittig oder rechts innerhalb eines Rechtecks angegeben werden.

Als linke obere Ecke des unsichtbaren Rechtecks gilt die aktuelle Grafikcursorposition. Mittels Bezugspunkt und den beiden Offsets wird die untere rechte Ecke angegeben. Eine Ausgabe in den RastPort erfolgt nicht.

WVICMD_SETAFPT : AreaPtSz, AreaPtrn;

Um für Flächenfüllkommandos beliebig Raster verwenden zu können, wird diese Funktion eingesetzt.

Wurde der Wert AreaPtrn mit einer -1L angegeben, dann muß das Tag WVIA_AreaPtrn die richtige Rasteradresse enthalten. Die Dimension des Raster errechnet sich aus Zwei hoch ArePtSz (2^{AreaPtSz}).

Jetzt steht dieses Raster allen Kommandos zu Verfügung.

WVICMD_SNAPCURSOR : BPkt;

Schreibt die aktuelle Grafikkursorposition an den angegebenen Bezugspunkt zurück. Möglich sind diese im Bereich von (0 - 15).

WVICMD_SNAPX : BPkt;

Schreibt die aktuelle X-Position des Grafikkursor in den Bezugspunkt zurück.

WVICMD_SNAPY : BPkt;

Schreibt die aktuelle Y-Position des Grafikkursor in den Bezugspunkt zurück.

WVICMD_TAGMOVE : TPoint;

Setzt den Grafikkursor auf eine Position, deren Werte aus einem WVIA_TPoint-Tag entnommen werden. Dabei bestimmt TPoint die Nummer des Tags.

WVICMD_TAGIMAGE : BPKT, XOffset, YOffset, Type, VImageTags;

Position und Dimension errechnen sich wie bei WVICMD_IMAGE. Sehen Sie also dort noch einmal nach.

Als Imageadresse wird aber der im Tag WVIA_TagImage angegeben Wert genommen. VImageTags enthält dabei die Tags für den WZ_DrawVImage() - Aufruf. Wird dieser Wert mit -1L angegeben, dann werden die aktuellen Tags dafür genommen. Der Type für den Funktionsaufruf wird dem Tag WVIA_TagImageCode entnommen.

WVICMD_BITMAP_TO_RP : BPKT, XOffset, YOffset, Map;

Damit lassen sich Bitmaps in den RastPort blitten.

BPkt, XOffset und YOffset ergibt die linke obere Ecke, an die diese Bitmap kopiert wird. Map enthält die Nummer des Tags WVIA_BitMap. Die Ausdehnung der Bitmap muß mit den Tags WVIA_BitMapWidth und WVIA_BitMapHeight angegeben werden.

WVICMD_FILLBORDER : BPKT, XOffset, YOffset, HBorder, VBorder;

Diese Funktion zeichnet einen Rahmen mit der aktuellen APen. Gezeichnet wird von der linken oberen Ecke der gesamten Vektorgrafik bis zu deren unteren rechten

Ecke. Dabei wird ein Bereich von der aktuellen Grafik-cursorposition bis zu dem angegeben Bezugspunkt nicht überzeichnet. Bezugspunkt, XOffset und YOffset werden zur Bestimmung der rechten unteren Ecke genommen, deren Inhalt NICHT überzeichnet werden soll.

Mit HBorder und VBorder kann die Dimension der Vektor-grafikdimension innerhalb dieser Routine verringert werden.

WVICMD_Beep :

ruft die Funktion intuition.library/DisplayBeep mit dem Parameter Screen = NULL auf.

WVICMD_AREAINIT :

Initialisiert den RastPort, um Areafunktionen nutzen zu können. Das Tag WVIA_TmpRas muß einen Zeiger auf eine initialisierte TmpRas-Struktur enthalten.

WVICMD_AREAMOVE : BPKT, XOffset, YOffset;

Damit wird der Grafikkursor für Areafunktionen auf die Position des Bezugspunktes gesetzt.

Dabei werden XOffset und YOffset zur Position addiert. Hiernach dürfen 3 Aufrufe des Kommandos

WVICMD_AREADRAW folgen.

WVICMD_AREADRAW : BPKT, XOffset, YOffset;

Zieht eine Begrenzungslinie zum angegeben Bezugspunkt.

Dabei werden XOffset und YOffset zur Position addiert. Diese Funktion darf maximal 3mal hintereinander aufgerufen werden !

WVICMD_AREAEND :

Zeichnet ein ausgefülltes Image in den RastPort.

Dabei werden APen und das aktuelle Raster berücksichtigt.

Danach dürfen AreaMove und AreaDraw wieder erfolgen.

RESULT

success - TRUE in jedem Falle

1.23 wizard.library/WZ_EasyRequestArgs

NAME

WZ_EasyRequestArgs -- einfachen Requester handhaben

SYNOPSIS

result = WZ_EasyRequestArgs(surface, window, id, args)

D0 A0 A1 D0 A2

LONG WZ_EasyRequestArgs(
APTR, struct Window *, ULONG, void *);

FUNCTION

Ähnlich der EasyRequestArgsA-Funktion der intuition.library.

Dabei wird automatisch die Tastatur überwacht.

WARNING

Die übergebenen Argumente können nicht überprüft werden.

Nur eine Gadget-Anwahl mit der Maus oder der

Tastatur kann diesen Requester beenden. Der Userport wird nicht
gegen Eingaben gesperrt, also Locken Sie das Fenster irgendwie
selber!

INPUTS

surface - Returnwert von WZ_OpenSurface()

window - Adresse der Window-Struktur, dessen Fenster gesperrt
werden soll

id - Nr des Requesters aus dem StormWizard

args - Adresse der Argumentenliste

RESULT

result - Nummer des Gadgets, der zum verlassen des Requesters
geführt hat oder -1, wenn der Requester aus

Speicherplatzgründen nicht dargestellt werden konnte.

SEE ALSO

[WZ_InitEasyStruct\(\)](#)

1.24 wizard.library/WZ_FreeWindowHandle

NAME

WZ_FreeWindowHandle -- WizardWindowHandle freigeben

SYNOPSIS

WZ_FreeWindowHandle(winhandle)

VOID WZ_FreeWindow(struct WizardWindowHandle *);

FUNCTION

Gibt den Speicher und damit alle Objecte die zu einem
Fenster gehören frei. Ein geöffnetes Fenster wird mit
diesem Funktionsaufruf gleichzeitig geschlossen.

INPUTS

winhandle - WizardWindowHandle von WZ_AllocWindowHandle

SEE ALSO

[WZ_AllocWindowHandleA\(\)](#)

1.25 wizard.library/WZ_GadgetConfig

NAME

WZ_GadgetConfig -- Gadgetconfigstring holen

SYNOPSIS

```
string = WZ_GadgetConfig(winhandle, gadget)
```

D0 A0 A1

STRPTR WZ_GadgetConfig(

struct WizardWindowHandle *, struct Gadget *);

FUNCTION

Der Configurationsstring zu einem Gadget kann mittels dieser Funktion geholt werden. Dieser wurde im StormWizzard eingegeben.

INPUTS

winhandle - Zeiger auf ein WizardWindowHandle

gadget - Zeiger auf ein Wizard-Gadget

RESULT

string - Stringadresse oder NULL in Fehlerfall

SEE ALSO

[WZ_MenuConfig\(\)](#), [WZ_GadgetHelp\(\)](#)

1.26 wizard.library/WZ_GadgetHelp

NAME

WZ_GadgetHelp -- Gadgethilfestring holen

SYNOPSIS

```
string = WZ_GadgetHelp(winhandle, iaddress)
```

D0 A0 A1

STRPTR WZ_GadgetHelp(struct WizardWindowHandle *, APTR);

FUNCTION

Der Hilfestring zu einem Gadget oder Fenster kann mittels dieser Funktion geholt werden. Dieser wurde im StormWizzard eingegeben.

Wurde die Oberflächenbeschreibung mit dem Tag SFH_Catalog bereitgestellt, dann wird der übersetzte String geliefert.

WARNING

Ist das übergebene Gadget nicht als Window im WizardWindow-Handle eingetragen, wird ohne Überprüfung davon ausgegangen, das es sich um ein Gadget der Wizard.library handelt.

INPUTS

winhandle - Zeiger ein WizardWindowHandle
iaddress - Zeiger auf ein Wizardgadget oder ein
geöffnetes Fenster

RESULT

string - Stringadresse oder NULL in Fehlerfall

SEE ALSO

WZ_MenuHelp(), WZ_GadgetConfig()

1.27 wizard.library/WZ_GadgetHelpMsg

NAME

WZ_GadgetHelpMsg -- Gadgethelpmessage holen

SYNOPSIS

succes = WZ_GadgetHelpMsg(winhandle, winhaddress, iaddress
mousex, mousey, flags)

D0 A0 A1 A2

D0 D1 D2

BOOL WZ_GadgetHelpMsg(struct WizardWindowHandle *,
struct WizardWindowHandle **,
APTR *, WORD, WORD, UWORD);

FUNCTION

Um unter V37 und V38 des Amiga-OS auch GadgetHelpMessages empfangen zu können, wurde diese Funktion geschrieben.

Wahlweise kann diese Funktion ab V39 zu einer DummyFunktion werden, um diese Nachrichten von Intuition zu bekommen.

WARNING

In späteren Version kann IAddress auch Objecte liefern, die zwar unter der Maus liegen und auch zur Surface gehören, nicht aber im aktiven Fenster liegen. Das schliesst Fenster natürlich selbst auch ein.

INPUTS

winhandle - Zeiger ein WizardWindowHandle, von dem
ein MausMove gemeldet wurde
winhaddress - Zeiger auf einen 4Byte großen Speicherbereich,
der von der Funktion genutzt wird, um einen
2.Wert zurückzuliefern.
iaddress - Zeiger auf einen 4Byte großen Speicherbereich,
der von der Funktion zur Rückgabe eines 3.Wertes

genutzt wird.

mousex - X-Pos. der Maus

mousey - Y-Pos. der Maus

flags - folgende sind definiert

WGHF_IgnoreOS - um auch unter 3.0 und höher diese Funktion nutzen zu können

WGHF_FullControl - um auch Nachrichten von anderen Fenstern empfangen zu können. Dieses Flag funktioniert unter V1.0 noch nicht !

RESULT

success - hat sich die Position der Maus geändert und damit auch das darunterliegende Object, dann ist dieser Wert TRUE, andernfalls FALSE.

winhaddress - Wenn success gleich TRUE ist, dann steht hier ein Zeiger auf das WizardWindowHandle, welches sich unter der Maus befindet.

iaddress - Wenn success gleich TRUE ist, dann steht hier ein Zeiger auf das darunterliegende Object.

Er muß dann genauso wie das entsprechende Feld einer IntuitMessage bei einer IDCMP_GADGETHELP-Msg des Betriebssystems bewertet werden.

SEE ALSO

[WZ_GadgetHelp\(\)](#), [WZ_MenuHelp\(\)](#),

1.28 wizard.library/WZ_GadgetKey

NAME

WZ_GadgetKeyA -- Gadgets über Tastendruck informieren

WZ_GadgetKey -- Gadgets über Tastendruck informieren

SYNOPSIS

success = WZ_GadgetKeyA(winhandle, code, qualifier, tags)

D0 A0 D0 D1 A1

BOOL WZ_GadgetKeyA(

struct WizardWindowHandle *, ULONG, ULONG,

struct TagItem *);

success = WZ_GadgetKey(

winhandle, code, qualifier, firstTag, ...)

BOOL WZ_GadgetKey(

struct WizardWindowHandle *, ULONG, ULONG, APTR, Tag, ...);

FUNCTION

Um die Gadgets systemkonform über einen Tastendruck zu informieren, sollte man diese Funktion benutzen.

Ist ein Stringgadget oder ein Integergadget für diese Tastenkombination zuständig, dann wird dieses hier automatisch aktiviert. Sollte ein anderes Gadget für die Tastenkombination vorbelegt sein, dann sendet diese eine neue IDCMP_IDCMPUPDATE - Message an das Fenster, dadurch werden Notifys automatisch berücksichtigt !

WARNING

Wenn mehrere Gadgets für diese Tastenkombination verantwortlich sind und gleichzeitig sichtbar sind, dann ist nicht vorhersagbar welches Gadget die Message auslöst.

INPUTS

winhandle - WizardWindowHandle, in dessen Fenster diese Taste gedrückt wurde

key - ASCII-Tastencode der gedrückten Taste

qualifier - Qualifercodex der bei der Taste gegohten hat

RESULT

success - war ein Gadget für diese Kombination zuständig dann TRUE, ansonsten FALSE

1.29 wizard.library/WZ_GetNode

NAME

WZ_GetNode -- Node besorgen

SYNOPSIS

```
node = WZ_GetNode(list, number)
```

D0 A0 D0

```
struct WizardTreeNode *WZ_GetNode(struct MinList *, UWORD);
```

FUNCTION

Diese Funktion ist für die Hierarchy-Klasse geschrieben worden.

Mit ihr kann die sichtbare Node des Baumes geholt werden, das sich unter dem Cursor befindet. Rufen Sie diese Funktion also mit dem Wert aus dem Tag WHIERARCHYA_Selected auf.

WARNING

In der jetzigen Version der Library muß es sich um eine Liste handeln, deren Nodes als Baum verkettet sind.

INPUTS

list - Adresse der List-Struktur

number - Nummer der sichtbaren Node, deren Adresse geholt werden soll

RESULT

node - Adresse der WizardTreeNode oder NULL im Fehlerfall

SEE ALSO

[WZ_ListCount\(\)](#)

1.30 wizard.library/WZ_InitEasyStruct

NAME

WZ_InitEasyStruct -- Easystruct initialisieren

SYNOPSIS

```
easy = WZ_InitEasyStruct(surface,easystruct,id,size)
```

D0 A0 A1 D0 D1

```
struct EasyStruct *WZ_InitEasyStruct(  
APTR,struct EasyStruct *, ULONG *, ULONG, ULONG);
```

FUNCTION

Initialisiert die angegebene EasyStruct-Struktur.

WARNING

INPUTS

surface - der Returnwert von WZ_OpenSurface()

easystruct - Zeiger auf eine EasyStruct der intuition.library

id - Nummer des Requesters aus dem StormWizard

size - Größe dieser EasyStruct-Struktur

RESULT

easy - Adresse der übergebenen EasyStruct-Struktur
oder im Fehlerfall NULL

SEE ALSO

[WZ_EasyRequestArgs\(\)](#)

1.31 wizard.library/WZ_ListCount

NAME

WZ_ListCount -- Anzahl der sichtbaren Nodes holen

SYNOPSIS

```
count = WZ_ListCount(list)
```

D0 A0

```
ULONG WZ_ListCount(struct MinList *);
```

FUNCTION

Diese Funktion ist für die Hierarchy-Klasse geschrieben worden.

Mit ihr kann die Anzahl der sichtbaren Nodes dieses Baumes geholt werden.

WARNING

Es muß sich um eine Liste, in der WizardTreeNode's verkettet sind.

INPUTS

list - Adresse der List-Struktur

RESULT

count - Anzahl der sichtbaren Nodes

SEE ALSO

[WZ_GetNode\(\)](#)

1.32 wizard.library/WZ_LockWindow

NAME

WZ_LockWindow -- Fenster sperren

SYNOPSIS

WZ_LockWindow(winhandle)

A0

VOID WZ_LockWindow(struct WizardWindowHandle *);

FUNCTION

Diese Funktion sperrt das Fenster für alle möglichen Formen der Benutzereingabe. Gleichzeitig wird der Mauszeiger umgestellt.

INPUTS

winhandle - WizardWindowHandle-Struktur, dessen Fenster gesperrt werden soll

SEE ALSO

[WZ_UnlockWindow\(\)](#)

1.33 wizard.library/WZ_LockWindows

NAME

WZ_LockWindows -- alle Fenster sperren

SYNOPSIS

WZ_LockWindows(surface)

A0

VOID WZ_LockWindows(APTR);

FUNCTION

Diese Funktion führt für alle Fenster ein WZ_LockWindow() durch, die zu einer Oberflächenbeschreibung gehören.

INPUTS

surface - Returnwert von WZ_OpenSurface()

SEE ALSO

WZ_UnlockWindows()

1.34 wizard.library/WZ_MenuConfig

NAME

WZ_MenuConfig -- Menüconfigstring holen

SYNOPSIS

string = WZ_MenuConfig(winhandle, code)

D0 A0 D0

STRPTR WZ_MenuConfig(struct WizardWindowHandle *, ULONG);

FUNCTION

Der Configurationsstring zu einem Menü kann mittels dieser Funktion geholt werden. Dieser wurde im StormWizzard festgelegt.

INPUTS

winhandle - Zeiger auf ein WizardWindowHandle

code - MenuCode des angewählten MenuTitles,-item

oder -subitem

RESULT

string - Stringadresse oder NULL in Fehlerfall

SEE ALSO

WZ_GadgetConfig(), WZ_MenuHelp()

1.35 wizard.library/WZ_MenuHelp

NAME

WZ_MenuHelp -- Menühilfestring holen

SYNOPSIS

string = WZ_MenuHelp(winhandle, code)

D0 A0 D0

STRPTR WZ_MenuHelp(struct WizardWindowHandle *, ULONG);

FUNCTION

Der Hilfestring zu einem Menüpunkt kann mittels dieser Funktion geholt werden. Dieser wurde im StormWizzard festgelegt.

Wurde die Oberflächenbeschreibung mit dem Tag SFH_Catalog bereitgestellt, dann wird der übersetzte String geliefert.

INPUTS

winhandle - Zeiger auf ein WizardWindowHandle

code - MenuCode des angewählten MenuTitles,-item

oder -subitem

RESULT

string - Stringadresse oder NULL in Fehlerfall

SEE ALSO

[WZ_GadgetHelp\(\)](#), [WZ_MenuConfig\(\)](#)

1.36 wizard.library/WZ_NewObjectA

NAME

WZ_NewObjectA -- Object anlegen

WZ_NewObject -- variable Parameterübergabe für Hochsprachen-programmierer

SYNOPSIS

obj = WZ_NewObjectA(class, tags)

D0 D0 A0

APTR WZ_NewObjectA(ULONG, struct TagItem *);

obj = WZ_NewObject(class, firstTag, ...)

APTR WZ_NewObject(ULONG, Tag, ...);

FUNCTION

Legt ein Object einer angegebenen Klasse an.

INPUTS

class - Klasse von der ein Object angelegt werden soll

tags - folgende Tags sind definiert:

GA_DrawInfo, struct DrawInfo *

Geben Sie hier IMMER einen Zeiger auf einen DrawInfo an.

WGA_Label -- (V1) [C...], STRPTR

Übergibt einen Zeiger auf einen String, der für das

Anlegen des Objects wichtig ist. Es handelt sich hierbei

um ein Universal-Tag.

WGA_Label2 -- (V1) [C...], STRPTR

Dient dazu einen zweiten Zeiger für das Anlegen eines

Objektes übergeben zu können. Universal-Tag !

WGA_TextFont -- (V1) [C...], struct TextFont *

Mit diesem Tag kann der von DrawInfo vorgegebene

Zeichensatz überschrieben werden. Damit ist es möglich jedem Object einen eigenen Zeichensatz zu geben.

Universal-Tag ! (Vorgabe: DrawInfo->Font)

WGA_Flags -- (V1) [C...], UWORD

Hier können Sie Flags für das zu erzeugende Object angeben. folgenden allgemeine Flags sind definiert :

WGF_GadgetHelp -- falls dieses Gadget eine GadgetHelp Message senden soll, wenn es von der Maus überfahren wird.

WGF_Disabled -- falls dieses Gadget nicht anwählbar sein soll

WGA_Priority -- (V1) [C.G.], UBYTE

Da es sinnvoll ist, jedem Object einen eigene Priorität zu geben, wurde dieses Tag ins Leben gerufen. Geben Sie hier einen Wert im Bereich von 0 bis 255 an. (Vorgabe 0).

WGA_RelHeight -- (V1) [C...], UBYTE

Geben Sie hier den oberen und unteren Rand eines Objectes von seiner TextDimension aus an. (Vorgabe 2).

WGA_MinWidth -- (V1) [C.G.], UWORD

Dieses Attribut kann von jedem Object abgefragt werden. Hiermit kann für anzulegende Objecte die minimale Breite angegeben werden. Dieses Tag ist Klassenabhängig.

WGA_MinHeight -- (V1) [C.G.], UWORD

Dieses Attribut kann von jedem Object abgefragt werden. Hiermit kann für anzulegende Objecte die minimale Höhe angegeben werden. Dieses Tag ist stark Klassenabhängig.

WGA_Link -- (V1) [..S.], struct Gadget *

Um Kommunikationen zwischen Gadgets zu erlauben, deren Inhalt völlig unwichtig für den Programmierer ist, sollte dieses Tag verwendet werden.

WGA_LinkData -- (V1) [C...], UBYTE

Wenn Links zwischen Gadgets gelegt sind, dann ist es klassenabhängig, ob das Object (welches den Link bekommt) auch eine Zeilenangabe erhalten muß. Geben Sie ganz einfach die Zeilennummer an, wenn in Richtung Labelgadget gelinkt wird. Für die Tastatursteuerung ist dieses Tag unerlässlich.

WGA_HelpText -- (V1) [C...], STRPTR

Hilfetexte, die sich mittels **WZ_GadgetHelp()** von einem

Wizardgadget abfragen lassen sollen.

WGA_Config -- (V1) [C...], STRPTR

Die Function **WZ_GadgetConfig()** gibt diesen String zurück.

Geben Sie hier also diesen String mit einem NullByte abgeschlossen an.

WGA_NewImage -- (V1) [C...], struct WizardNewImage *

Damit dieses Object sein Image bekommt, das es für die Darstellung und seine Dimensionsberechnung benötigt, ist dieses Tag definiert worden.

WGA_SelNewImage -- (V1) [C...], struct WizardNewImage *

Geben Sie hier eine WizardNewImage-Beschreibung an, damit sich das Object im selektierten Zustand richtig zeichnen kann.

WGA_Group -- (V1) [C...], struct Gadget *

Damit sich das anzulegende Object in die richtige Gruppe einhängen kann, ist dieses Tag notwendig.

Übergeben Sie die Gadget-Struktur des Gruppengadgets.

WGA_GroupPage -- (V1) [C...], UWORD

Falls das Gruppengadget mehrere Seiten besitzt, dann müssen Sie hier die Seite angeben, in die unser anzulegendes Objekt eingehangen werden soll. (Vorgabe 0)

WGA_Locale -- (V1) [C...], struct Locale *

Bis jetzt ist dieses Tag nur bei Erzeugen eines Datumsgadgets sinnvoll, um die Tageskürzel in der richtigen Sprache zu erhalten. Dieses Tag muß angegeben werden.

WCLASS_LAYOUT:

- starten und regeln den Layoutvorgang

WLAYOUTA_RootGadget -- (V1) [C...], struct Gadget *

Rootgruppe, deren Layoutvorgang gestartet werden soll

WLAYOUTA_Type -- (V1) [C...], UWORD

Art des Rootgadgets, noch nicht unterstützt

WLAYOUTA_BorderLeft -- (V1) [C...], WORD

WLAYOUTA_BorderRight -- (V1) [C...], WORD

WLAYOUTA_BorderTop -- (V1) [C...], WORD

WLAYOUTA_BorderBottom -- (V1) [C...], WORD

Borderwerte, die beim Layoutvorgang freigehalten werden sollen. (Vorgabe 0)

WLAYOUTA_StackSwap -- (V1) [C...], struct StackSwapStruct *

Gleichnamige Struktur, in der der Stack für den Layoutvorgang festgehalten ist. Dieses Tag muß angegeben werden.

WCLASS_HGROUP und WCLASS_VGROUP:

- machen Paging (max. 32)
- berechnen Dimension und Position ihrer Mitglieder
- Flags: WGRPF_EqualSize, um allen Mitgliedern die selbe Mindestdimension zu verordnen und WGRPF_DockMode, um es in den Dockmodus umzuschalten.

WGROUPA_ActivePage -- (V1) [CSGN], UWORD

Dieses Tag beschreibt, welche Seite dieser Gruppe sichtbar ist.

WGROUPA_MaxPage -- (V1) [C...], UWORD

Gibt die maximale Seitennummer an, welche von diesem Gadget verwaltet werden soll. Maximalwert ist 31.

Dieses Tag wird überschrieben, falls der GruppenTitel mehrzeilig ist.

WGROUPA_HBorder -- (V1) [C...], WORD

Dieser Wert bestimmt den unsichtbaren linken und rechten Rand, welcher freigelassen werden soll.

WGROUPA_VBorder -- (V1) [C...], WORD

Ähnlich dem HBorder wird hiermit der obere und untere Rand angegeben. Falls ein einzeliger Title übergeben wurde, dann wird zum oberen Rand noch die Zeichensatzhöhe hinzuaddiert.

WGROUPA_BHOffset -- (V1) [C...], WORD

Bestimmt den Offset vom inneren Bereich nach links und rechts weg, mit dem ein angegebenes Frame, gezeichnet wird.

WGROUPA_BVOffset -- (V1) [C...], WORD

Hiermit legt man den Offset vom inneren Bereich nach oben und unten fest, mit dem ein eventuelles Frame gezeichnet werden soll.

WGROUPA_Space -- (V1) [C...], UWORD

Legt den festen Mindestabstand fest, der zwischen den Mitgliedern dieser Gruppe vorhanden sein soll.

Besitzt diese Gruppe kein oder nur ein Mitglied, dann ist dieses Tag wirkungslos.

WGROUPA_VarSpace -- (V1) [C...], UWORD

Bestimmt das Verhältnis, mit dem der Platz auf die Mitglieder und deren Zwischenräume verteilt werden soll. Maximalwert ist 65535 und bedeutet, daß der Platz voll an die Zwischenräume verteilt werden soll. Alle Objecte dieser Gruppe besitzen dann ihre minimale Dimension. Ein Wert von 32768 veranlasst das Gruppengadget das Verhältnis zwischen genutzen Platz und Spaceraum etwa gleich zu berechnen. Dies ist natürlich nicht immer möglich.

WGROUPA_FrameType -- (V1) [C...], UWORD

Bestimmt das zu zeichnende Frame im Bereich von 0 bis 8.
(Vorgabe 0).

WCLASS_BUTTON

- einfache Knöpfe

WBUTTONA_Label -- (V1) [CS..], STRPTR

Dient der Angabe eines Labels für ein Buttongadget, es überschreibt das Universaltag WGA_Label !

WCLASS_STRING

- wenn es mit Taste aktivierbar sein soll, muß es mit einem Label gelinkt werden

- Flag: WGF_KeyControl für TABCYCLE. Muß gesetzt sein !

WSTRINGA_String -- (V1) [CSGN], STRPTR

Übergibt die Adresse eines mit NullByte terminierten Strings. Dieses Tag überschreibt beim Anlegen eines Objektes das Universaltag WGA_Label.

WSTRINGA_MaxChars -- (V1) [C...], UWORD

Bestimmt die maximale Länge eines bearbeitbaren Strings ohne NullByte. Maximalwert ist 255.

WSTRINGA_Justification -- (V1) [C...], ULONG

Justierung (GACT_...). (Vorgabe Links).

WCLASS_CHECKBOX

- muß mit Labelgadget gelinkt werden.

WCHECKBOXA_Checked -- (V1) [CSGN], BOOL

Ist True bei einem eingedrückten Zustand, sonst FALSE.

WCLASS_MX

- muß mit Labelgadget gelinkt werden.

WMXA_Active -- (V1) [CSGN], UWORD

Beschreibt den ausgewählten Knopf innerhalb dieses Objekts.

WCLASS_LABEL

- häufig als Zielobjekt für Links

- Texte können mehrzeilig sein

- sendet keine Notifys

WLABELA_FrameType -- (V1) [C...], UWORD

Kennzeichnet das Frame, das um dieses Objekt herum gezeichnet werden soll.

WLABELA_Space -- (V1) [C...], UWORD

Abstand in Bildpunkten zwischen den Textzeilen.

WLABELA_BGPen -- (V1) [C...], UWORD

Dieses beschreibt die Hintergrundfarbe. (Vorgabe 0)

WLABELA_TextPlace -- (V1) [C...], UWORD

Ausrichtung der Textzeilen. Geben Sie hier folgendes an:

WZRDPLACE_LEFT, WZRDPLACE_CENTER oder WZRDPLACE_RIGHT.

WLABELA_HighLights -- (V1) [C...], ULONG

Beschreibt, welche Zeilen bei der Darstellung in heller Textfarbe dargestellt werden sollen. um Zeile 0 und Zeile 2 hell darzustellen, muß eine $2^0 + 2^2 = 5$ angegeben werden.

WLABELA_Styles -- (V1) [C...], ULONG

Beschreibt, welche Zeilen bei der Darstellung in mit dem TextStyle Fett dargestellt werden soll.

Um Zeile 0 und Zeile 2 fett darzustellen, muß eine 5 angegeben werden.

WLABELA_Label -- (V1) [C...], STRPTR

Adresse des darzustellenden Textes, dieses Tag überschreibt WGA_Label.

WLABELA_Lines -- (V1) [..G.], UWORD

Anzahl der Textzeilen dieses Objects.

WCLASS_INTEGER

- ermöglicht Eingabe von Zahlen in 3 Zahlensystemen

- Flag: WGF_KeyControl für TABCYCLE. Muß gesetzt sein !

WINTERGERA_Long -- (V1) [CSGN], LONG

vorzeichenbehafter Zahlenwert der vom Benutzer verändert werden kann.

WINTERGERA_MinLong -- (V1) [C...], LONG

WINTERGERA_MinLong -- (V1) [C...], LONG

Begrenzung des Gadgetwertes.

WINTERGERA_Justification -- (V1) [C...], ULONG

Justierung (GACT_...). (Vorgabe Links).

WCLASS_HSCROLLER und WCLASS_VSCROLLER

- Schieberegler für das Scrollen von Listen.

- Links möglich in Richtung:

ListView, MultiListView oder Hierarchy

- Flag WSCF_NewLook, um den Regler im neuen Look zu sehen.

WSCOLLERA_Top -- (V1) [CSGN], LONG

Zahlenwert, der sich im Normalfall im Bereich von 0 bis Total-1 bewegt.

(Vorgabe 0)

WSCOLLERA_Visible (V1) [CSG.], ULONG

Bestimmt die Anzahl der sichtbaren Einträge innerhalb einer möglichen Liste.

WSCOLLERA_Total -- (V1) [CSG.], ULONG

Anzahl der Einträge einer möglichen Liste.

(Vorgabe 10).

WCLASS_ARROW

- Pfeilgadgets

- Links in Richtung Proportionalgadgets möglich, dann senden die Proportionalgadgets eine Message

WARROWA_Type -- (V1) [C...], UWORD

Pfeilart im Bereich von 0 bis 3. (Links, Rechts, Hoch und Runter). (Vorgabe 0 - Links)

WARROWA_Step -- (V1) [CS..], UWORD

Impulsgröße, bei Wiederholung eines Steps infolge eines Ticks

WCLASS_LISTVIEW und WCLASS_MULTILISTVIEW

- stellt eine Liste mit WizardNodes dar

- Flag WLVF_ReadOnly, um den Rahmen eingedrückt erscheinen zu lassen und Flag WLVF_DoubleClicks, um deren Nachrichten zu

erhalten. s.u.

WLISTVIEWA_Top -- (V1) [CSG.], UWORD

obere Position, mit der die Ausgabe der Liste erfolgen soll. (Vorgabe 0)

WLISTVIEWA_SBPen -- (V1) [CS.], UWORD

Farbe, mit der der Cursor einer selektierten Liste dargestellt wird. (Vorgabe WZRD_FILLPEN)

WLISTVIEWA_Selected -- (V1) [CSGN], WORD

Nummer des selektierten Eintrages in der Liste.

Bei MultiListView handelt es sich um den Eintrag, bei dem sich die Selektierung ändert.

WLISTVIEWA_List -- (V1) [CS.], struct MinList *

Adresse der darzustellenden Liste, in der Wizard-ListNodes verkettet sind. Dabei können Sie den Strings eine Vektorgrafik vorangestellt. Geben Sie in einem solchen Falle die Adresse in der Node an.

Ist eine Node nicht selektiert, dann erfolgt der WZ_DrawVImage() - Aufruf dem type = 0, ansonsten mit eins. Bei einem MultiListView müssen Sie dem Flag WLNF_Selected entnehmen, ob diese Node selektiert wurde bzw. ist. Das Flag LNF_Selected darf bei einem ListView nicht verwendet werden !

WLISTVIEWA_Offset -- (V1) [CS.], UWORD

Offset, der angibt, ab welchen Buchstaben die Strings einer Liste dargestellt werden sollen. Bitte verwenden Sie dieses Tag niemals, wenn die Nodes Vektorgrafiken besitzen.

WLISTVIEWA_DoubleClick -- (V1) [...N], BOOL

Wurde eine Message von einem ListView abgeschickt und dieses Tag enthält TRUE, dann handelte es sich um einen Doppelklick als Auslöser.

WLISTVIEWA_Visible -- (V1) [.S.], UWORD

Geben Sie hier die Nummer einer Node an, die in jedem Fall sichtbar sein soll. Diese Tag kommt in Konflikt mit dem Tag WLISTVIEWA_Top, da es, wenn die Node nicht sichtbar ist, diesen Wert modifiziert.

WLISTVIEWA_ImageWidth -- (V1) [CS.], UWORD

Breite in Pixeln, mit der Images gezeichnet werden sollen.

WLISTVIEWA_HeaderSpace

WLISTVIEWA_HeaderString

WLISTVIEWA_HeaderStyle

Diese Tags bekommen erst bei späteren Versionen der Library eine Bedeutung.

WCLASS_TOGGLE

- ähnlich den CheckBoxgadgets
- Flag WTGF_SimpleMode, wenn sich das Object nur selektieren lassen soll, das Deselektieren ist dann nicht mehr möglich

WTOGGLEA_Checked -- (V1) [CSGN], BOOL

Ist True bei einem eingedrückten Zustand, sonst FALSE.

WCLASS_LINE

- stellen Linien dar
- sendet keine Notifys

WLINEA_Type -- (V1) [C...], UWORD

Beschreibt, wie das Lineagadget eine Linie zu zeichnen hat.

Diese sind horizontal(0), vertikal(1), selektiert horizontal(2) und selektiert vertikal(2). (Vorgabe 0)

WLINEA_Label -- (V1) [C...], STRPTR

Übergibt die Adresse eines mit NullByte terminierten Strings, der bei horizontalem Type mit in die Linie hineingeschrieben wird. Dieses Tag überschreibt das UniversalTag WGA_Label.

WCLASS_COLORFIELD

- kann Farben für eine Legende darstellen
- muß mit einem Labelgadget gelinkt werden
- sendet keine Notifys

WCOLORFIELDA_Pen -- (V1) [CS..], UWORD

Nummer des Farbregisters, deren Farbe in dem umrahmten Kästchen erscheinen soll. Es sind die WZRD_... - Farben zugelassen ! (Vorgabe 0)

WCLASS_ARGS

- kann bis zu 10 verschiedene Argumente darstellen
- sendet keine Notifys

WARGSA_Format -- (V1)[.S..], STRPTR

Format, falls das im StormWizard angegebene nicht beibehalten werden soll.

WARGSA_TextPlace -- (V1) [C...], UWORD

Konstante die beschreibt, wie der formatierte Text innerhalb der grafischen Abmessungen justiert werden soll.

WARGSA_FrameType -- (V1) [C...], UWORD

Eine Zahl, die den zu zeichnenden FrameType beschreibt.

WARGSA_Arg0-9 -- (V1) [CS..], ULONG

Geben Sie hier ihr Argument an, welches dann nach dem übergebenen Formatierungsstring formatiert wird.

ACHTUNG: Wird die Adresse eines Strings übergeben, dann kann dieser nicht in einen eigenen Puffer kopiert werden !

WCLASS_GAUGE

- zum Darstellen von zeitlichen Abläufen.

- sendet keine Notifys

WGAUGEA_Total -- (V1) [CS..], UWORD

obere Grenze des Füllstandes

WGAUGEA_Current -- (V1) [CS..], UWORD

momentane Position des Füllstandes, dieser Wert darf sich im Bereich von 0 bis Total bewegen.

WGAUGEA_Format -- (V1) [CS..], STRPTR

Dieses Tag überschreibt WGA_Label und dient als Formatierungsstring für die Funktion der zusammen mit den Argumenten an die exec.library/RawDoFmt() übergeben wird. Dieses Tag überschreibt das UniversalTag WGA_Label.

WCLASS_CYCLE

- können für die Tastaturkontrolle mit einem Label gelinkt werden

WCYClea_Active -- (V1) [CSGN], UWORD

beschreibt den aktiven Eintrag.

WCYClea_Labels -- (V1) [CS..], STRPTR

Adresse mehrerer Strings, die mit einem Return ("\n") getrennt sind. Dieses Tag überschreibt das Tag WGA_Label.

WCLASS_VECTORBUTTON

- Verhalten sich wie Objecte der Buttonklasse

WVECTORBUTTONA_Type -- (V1) [C...], UWORD

Nummer des Vektorimages, das für die Darstellung genommen werden soll: File(0), Drawer(1) oder Popup(2).

WCLASS_DATE

- Verhalten sich wie Kalenderblätter

WDATEA_Day -- (V1) [CSGN], UWORD

der selektierte Tag, der mindestens vom Wert Eins sein muß. (Vorgabe 1)

WDATEA_Month -- (V1) [CS..], UWORD

der Monat der für die Darstellung des Kalenderblattes wichtig ist. Mindestens eine Eins für den Januar !
(Vorgabe 1)

WDATEA_Year -- (V1) [CS..], UWORD

das Jahr, das mindestens 1978 betragen muß.
(Vorgabe 1978)

WCLASS_SPACE

- dienen als Platzhalter und Flächenfüller

WSPACEA_Pen -- (V1) [C...], UWORD

Farbe, die mit einem Raster gezeichnet werden soll.
Als zweite Farbe dient die Hintergrundfarbe.

WCLASS_IMAGE

- stellen Bitmaps im Fenster dar

- senden keine Notifys

WIMAGEA_BGPen -- (V1) [C...], UWORD

Farbe, mit der der eventuell vorhandene Rand dargestellt werden soll.

WIMAGEA_FrameType -- (V1) [C...], UWORD

Beschreibt, mit welchem Frame die Dimension des Objektes gezeichnet werden soll. Zulässig sind: WZRDFRAME_NONE, WZRDFRAME_ICON, WZRDFRAME_BUTTON, WZRDFRAME_STRING und WZRDFRAME_DOUBLEICON. (Vorgabe WZRDFRAME_ICON)

WIMAGEA_HBorder -- (V1) [C...], UWORD

Mindestabstand zum linken und rechten Rand. (Vorgabe 0)

WIMAGEA_VBorder -- (V1) [C...], UWORD

Mindestabstand zum oberen und unteren Rand. (Vorgabe 0)

WIMAGEA_NewImage -- (V1) [C...], struct WizardNewImage *

Adresse der WizardNewImage-Struktur, die das zu zeichnende

Image beschreibt. Dieses Tag überschreibt WGA_NewImage.

WCLASS_IMAGEBUTTON

- verhalten sich wie Buttons, sehen aber aus wie Images

WIMAGEBUTTONA_BGPen -- (V1) [C...], UWORD

Farbe, mit der der vorhandene Rand gezeichnet werden soll,
wenn das Image im Normalzustand ist.

WIMAGEBUTTONA_SelBGPen -- (V1) [C...], UWORD

Farbe, mit der der vorhandene Rand gezeichnet werden soll,
wenn das Image im selektierten Zustand ist.

WIMAGEBUTTONA_FrameType -- (V1) [C...], UWORD

Beschreibt, mit welchem Frame die Dimension des Objektes
gezeichnet werden soll. Zulässig sind: WZRDFRAME_NONE,
WZRDFRAME_ICON, WZRDFRAME_BUTTON, WZRDFRAME_STRING und
WZRDFRAME_DOUBLEICON. (Vorgabe WZRD_ICON)

WIMAGEBUTTONA_HBorder -- (V1) [C...], UWORD

Mindestabstand zum linken und rechten Rand. (Vorgabe 0)

WIMAGEBUTTONA_VBorder -- (V1) [C...], UWORD

Mindestabstand zum oberen und unteren Rand. (Vorgabe 0)

WIMAGEBUTTONA_NewImage -- (V1) [C...], struct WizardNewImage *

Adresse der WizardNewImage-Struktur, die das zu zeichnende
Image beschreibt. Dieses Tag muß angegeben werden.

WIMAGEBUTTONA_SelNewImage -- (V1) [C...], struct WizardNewImage *

Adresse der WizardNewImage-Struktur, die das selektierte
Image beschreibt.

WCLASS_IMAGETOGGLE

- verhalten sich wie Toggles, sehen aber aus wie Images

- Flag WITF_SimpleMode, wenn sich das Object nur selektieren lassen
soll, das Deselektieren ist dann nicht mehr möglich

WIMAGETOGGLEA_BGPen -- (V1) [C...], UWORD

Farbe, mit der der vorhandene Rand gezeichnet werden soll,
wenn das Image im Normalzustand ist.

WIMAGETOGGLEA_SelBGPen -- (V1) [C...], UWORD

Farbe, mit der der vorhandene Rand gezeichnet werden soll,
wenn das Image im selektierten Zustand ist.

WIMAGETOGGLEA_FrameType -- (V1) [C...], UWORD

Beschreibt, mit welchem Frame die Dimension des Objektes
gezeichnet werden soll. Zulässig sind: WZRDFRAME_NONE,

WZRDFRAME_ICON, WZRDFRAME_BUTTON, WZRDFRAME_STRING und
WZRDFRAME_DOUBLEICON. (Vorgabe WZRD_ICON)

WIMAGETOGGLEA_HBorder -- (V1) [C...], UWORD

Mindestabstand zum linken und rechten Rand. (Vorgabe 0)

WIMAGETOGGLEA_VBorder -- (V1) [C...], UWORD

Mindestabstand zum oberen und unteren Rand. (Vorgabe 0)

WIMAGETOGGLEA_NewImage -- (V1) [C...], struct WizardNewImage *

Adresse der WizardNewImage-Struktur, die das zu zeichnende

Image beschreibt. Dieses Tag muß angegeben werden.

WIMAGETOGGLEA_SelNewImage -- (V1) [C...], struct WizardNewImage *

Adresse der WizardNewImage-Struktur, die das selektierte

Image beschreibt.

WIMAGETOGGLEA_Checked -- (V1) [CSGN], BOOL

Ist True bei einem eingedrückten Zustand, sonst FALSE.

WCLASS_IMAGEPOPUP

- Images mit Textpopup-Effekt

- Flag WIPF_NewLook, um den Popup in den 3D-Look zu schalten

WIMAGEPOPUPA_BGPen -- (V1) [C...], UWORD

Farbe, mit der der vorhandene Rand gezeichnet werden soll,

wenn das Image im Normalzustand ist.

WIMAGEPOPUPA_FrameType -- (V1) [C...], UWORD

Beschreibt, mit welchem Frame die Dimension des Objektes

gezeichnet werden soll. Zulässig sind: WZRDFRAME_NONE,

WZRDFRAME_ICON, WZRDFRAME_BUTTON, WZRDFRAME_STRING und

WZRDFRAME_DOUBLEICON. (Vorgabe WZRD_ICON)

WIMAGEPOPUPA_HBorder -- (V1) [C...], UWORD

Mindestabstand zum linken und rechten Rand. (Vorgabe 0)

WIMAGEPOPUPA_VBorder -- (V1) [C...], UWORD

Mindestabstand zum oberen und unteren Rand. (Vorgabe 0)

WIMAGEPOPUPA_NewImage -- (V1) [C...], struct WizardNewImage *

Adresse der WizardNewImage-Struktur, die das zu zeichnende

Image beschreibt. Dieses Tag muß angegeben werden.

WIMAGEPOPUPA_TextPlace -- (V1) [C...], UWORD

Justierung für die Textausgabe während eines Popups.

WZRDPLACE_LEFT,...

WIMAGEPOPUPA_Labels -- (V1) [C...], STRPTR

Adresse der durch Return ("\n") getrennten Text für den

Popup-Dialog. Dieses Tag überschreibt das Tag WGA_Label.

WIMAGEPOPUPA_Selected -- (V1) [...N], UWORD

Nummer des Selektierten Texteintrages

WCLASS_TEXTPOPUP

- sehen aus wie Buttons, aber mit Popup-Effekt

- Flag WTPF_NewLook, um den Popup in den 3D-Look zu schalten

WTEXTPOPUPA_TextPlace -- (V1) [C...], UWORD

Justierung der Textausgabe bei einem Popup-Effekt

WTEXTPOPUPA_Labels -- (V1) [C...], STRPTR

Adresse der durch Return ("\n") getrennten Texte,
für den Popup-Dialog. Dieses Tag überschreibt das Tag

WGA_Label2 und muß angegeben werden.

WTEXTPOPUPA_Name -- (V1) [C...], STRPTR

Adresse des Textes, der in dem Button-Frame erscheinen
tut. Dieses Tag überschreibt WGA_Label !

WTEXTPOPUPA_Selectd -- (V1) [...N], UWORD

Nummer des selektierten Textes, bei einem Popup-Effekt

WCLASS_PALETTE

- ermöglichen Auswahl einer Farbe aus einer Farbpalette

WPALETTEA_Colors -- (V1) [C...], UWORD

Anzahl der Farben, die zur Auswahl stehen sollen oder
eine -1, um die maximale Farbanzahl zu nutzen.

Maximal darf dieser Wert 255 betragen.

WPALETTEA_Selected -- (V1) [CSGN], WORD

Nummer der selektierten Farbe. (Vorgabe -1)

WPALETTEA_Offset -- (V1) [C...], UWORD

Nummer des ersten zu Wahl stehenden Farbregisters.

WCLASS_VECTORPOPUP

- sehen aus wie Vectorbuttons, aber mit Popup-Effekt

- Flag WVPF_NewLook, um den Popup in den 3D-Look zu schalten

WVECTORPOPUPA_Type -- (V1) [C...], UWORD

Nummer des Vektorimages, das für die Darstellung genommen
werden soll: File(0), Drawer(1) oder Popup(2).

WVECTORPOPUPA_Labels -- (V1) [C...], STRPTR

Adresse der durch Return ("\n") getrennten Texte,
für den Popup-Dialog. Dieses Tag überschreibt das Tag

WGA_Label und muß angegeben werden.

WVECTORPOPUPA_TextPlace -- (V1) [C...], UWORD

Justierung der Textausgabe bei einem Popup-Effekt

WVECTORPOPUPA_Selected -- (V1) [...N]

Nummer des selektierten Textes, bei einem Popup-Effekt

WCLASS_HIERARCHY

- ähnlich den ListView

- können hierarchische Listen darstellen

- Flag WHRF_DoubleClicks erlaubt das Abschicken einer Nachricht
bei einem Doppelklick !

WHIERARCHYA_Image -- (V1) [.S..], struct WizardVImage *

ein Zeiger auf eine umfangreiches Vektorimages,
in dem alle Fälle definiert sind

WHIERARCHYA_ImageType -- (V1) [C...], UWORD

wählt ein bestehendes Image vor. Es kann sein:

Keins(0), Baum (1), Dreieck (2).

WHIERARCHYA_ImageWidth -- (V1) [CS..], UWORD

Breite der zu zeichnenden Images, die auch als
Schalter fungieren.

WHIERARCHYA_TreeImageWidth -- (V1) [C...], UWORD

Breite der Verbindungsstriche eines Baumes. Dieser
Wert dient zum Einrücken.

WHIERARCHYA_List -- (V1) [CS..], struct MinList *

Adresse der Liste, deren Inhalt dargestellt werden soll.

Dabei muß es sich um eine Liste mit WizardTreeNodees
handeln und deren Felder ParentNode und Childs müssen
korrekt initialisiert sein. Die Struktur kann und sollte
in Form eines Baumes verkettet sein. Alle direkt in der
MinList-Struktur verketteten Nodes müssen eine Null
im Feld ParentNode erhalten, da diese nicht existiert.

WHIERARCHYA_Visible -- (V1) [.S..], UWORD

Geben Sie hier die Nummer einer sichtbaren Node an,
die in jedem Fall im Fenster dargestellt werden soll.
Dabei zählen nur Nodes, deren Darstellung aufgrund der
offenen ElternNode möglich ist.

WHIERARCHYA_Selected -- (V1) [CSGN], WORD

Nummer des selektierten Eintrages in der Liste.

Bei MultiListView handelt es sich um den Eintrag, bei
dem sich die Selektierung ändert.

WHIERARCHYA_DoubleClick -- (V1) [...N], BOOL

Wurde eine Message von einem Object abgeschickt und dieses Tag enthält TRUE, dann handelte es sich um einen Doppelklick als Auslöser.

WHIERARCHYA_ImageHitTest -- (V1) [C...], APTR

Zeiger auf eine Routine, die überprüft, ob sich der Mausfeil innerhalb eines Schalters befindet.

Gibt Sie TRUE zurück, dann gilt der Schalter als angewählt. Folgenden Register werden übergeben:

D0 - X innerhalb der Grafik

D1 - Y innerhalb der Grafik

D2 - ImageWidth vom Tag WHIERARCHYA_ImageWidth

D3 - ItemHeight, im Normalfall Zeichensatzhöhe+1

A0 - (struct WizardVImage *)

WHIERARCHYA_SBGPen -- (V1) [CS..], UWORD

Farbe, mit der der Balken gezeichnet wird.

(Vorgabe WZRD_FILLPEN).

WHIERARCHYA_Top -- (V1) [.SG.], UWORD

Nummer der ersten darstellbaren Node.

WCLASS_HSLIDER und WCLASS_VSLIDER

- Schieberegler für alle möglichen Einstellung.

- Flag WSLF_NewLook, um den Regler im neuen Look zu sehen.

WSLIDERA_Level -- (V1) [CSGN], WORD

Zahlenwert, der sich im Bereich der Werte von Min bis Max bewegen kann.

(Vorgabe 0)

WSLIDERA_Min (V1) [CSG.], WORD

Bestimmt die untere Grenze, die vom Level-Wert nicht unterschritten werden darf. (Vorgabe -10)

WSLIDERA_Max -- (V1) [CSG.], WORD

Bestimmt die obere Grenze, die der Level-Wert nicht überschreiten darf. (Vorgabe 10)

RESULT

obj - Zeiger auf das erzeugte Object.

SEE ALSO

intuition.library/DisposeObject()

1.37 wizard.library/WZ_ObjectID

NAME

WZ_ObjectID -- ObjectID ermitteln

SYNOPSIS

```
success = WZ_ObjectID(surface,id,objectname)
```

D0 a0 a2 a1

FUNCTION

Diese Funktion liefert die ObjectID eines Objektes, dessen Name übergeben wurde.

INPUTS

surface - Returnwert von WZ_OpenSurface()

id - Zeiger auf einen 4Byte großen Speicherbereich, aus dem bei Erfolg die ID entnommen werden kann

objectname - Name des Objektes, dessen interne ID ermittelt werden soll

RESULT

sucess - wurde die ID gefunden, dann TRUE sonst FALSE

SEE ALSO

WZ_OpenSurface()

1.38 wizard.library/WZ_OpenSurfaceA

NAME

WZ_OpenSurfaceA -- Oberflächenbeschreibung verfügbar machen

WZ_OpenSurface -- variable Parameterübergabe für Hochsprachen-programmierer

SYNOPSIS

```
Surface = WZ_OpenSurfaceA(name, memaddr,tags)
```

D0 A0 A1 A2

```
APTR WZ_OpenSurfaceA(STRPTR, APTR,struct TagItem *);
```

```
Surface = WZ_OpenSurface(name, memaddr,firsttag, ... )
```

```
APTR WZ_OpenSurface(STRPTR, APTR,Tag, ... );
```

FUNCTION

Diese Funktion wird benutzt um eine Oberflächenbeschreibung benutzen zu können. Befindet sich die Datei bereits an einer Adresse im Speicher, kann der name mit NULL übergeben werden und memadr enthält die Adresse im Speicher.

Im Normalfall sollte der name den Zeiger auf einen Namen der

Datei enthalten und memaddr gleich NULL sein.

WARNING

Wird eine Adresse mittels memaddr übergeben, muß sichergestellt werden, das diese nur einmal bei einem WZ_OpenSurface - Aufruf initialisiert wird. D.h. Programme die von dieser Möglichkeit Gebrauch machen, dürfen nicht resident gemacht werden !

INPUTS

name - ein Zeiger auf den Namen der Datei (NullByte !)

memaddr - wenn name nicht angegeben wird, ein Zeiger auf eine Adresse, die die Beschreibung der Oberfläche enthält

tags - folgende Tags sind definiert:

SFH_Locale - (V1) , struct Locale *

ein Zeiger von locale.library/OpenLocale(),

falls die Standardeinstellung des Betriebssystems nicht verwendet werden soll.

SFH_Catalog - (V1), struct Catalog *

ein Zeiger auf den geöffnete Catalog,
siehe locale.library/OpenCatalog().

SFH_AutoInit - (V1), BOOL

soll die Oberflächenbeschreibung von einer vorgegebenen Speicheradresse geöffnet werden, dann kann das initialisieren hiermit abgeschaltet werden, falls bereits eine Initialisierung erfolgt ist. (Vorgabe TRUE)

RESULT

Surface - undefinierter Zeiger oder Null im Fehlerfall

SEE ALSO

[WZ_CloseSurface\(\)](#), [WZ_SnapShot\(\)](#)

1.39 wizard.library/WZ_OpenWindowA()

NAME

WZ_OpenWindowA -- Fenster öffnen

WZ_OpenWindow -- variable Parameterübergabe für Hochsprachenprogrammierer

SYNOPSIS

window = WZ_OpenWindowA(winhandle, newwin, tags)

D0 A0 A1 A2

```
struct Window *WZ_OpenWindowA(struct WizardWindowHandle *,
struct NewWindow *, struct TagItem *);
```

```
window = WZ_OpenWindow(winhandle, newwin, firstTag, ... )  
struct Window *WZ_OpenWindow(struct WizardWindowHandle *,  
struct NewWindow *, Tag, ... );
```

FUNCTION

Öffnet ein Fenster ähnlich dem Intuitionsaufruf.

WARNING

Das Benutzen der Intuition-Funktion führt zu Fehlern im Layout-vorgang.

INPUTS

winhandle - von WZ_AllocWindowHandle initialisiertes

WizardWindowHandle

newwin - NewWindowStruct or NULL

z.b. von WZ_CreateWindowObj()

tags - Tags (z.b. WA_AutoAdjust)

RESULT

window - Zeiger auf eine Window-Struktur

oder NULL im Fehlerfall

SEE ALSO

[WZ_CloseWindow\(\)](#)

1.40 wizard.library/WZ_SnapShot

NAME

WZ_SnapShot -- Fensterpositionen und -dimensionen fixieren

SYNOPSIS

```
success = WZ_SnapShot(surface)
```

D0 A0

```
BOOL WZ_SnapShot(APTR);
```

FUNCTION

Eine Funktion zum speichern der Fensterpositionen und -dimensionen in die .wizard - Datei zurück. Noch offene Fenster werden nicht berücksichtigt. Im allgemeinen sollte man diese Funktion am Ende des Program`s aufrufen.

WARNING

War die Funktion WZ_OpenSurface() mit dem memadr - Parameter aufgerufen worden, dann ist das fixieren nicht möglich.

INPUTS

surface - Returnwert von WZ_OpenSurface()

RESULT

success - TRUE -> wenn alles glatt ging,

FALSE -> bei einem Fehler

SEE ALSO

WZ_OpenSurface(), [WZ_CloseSurface\(\)](#)

1.41 wizard.library/WZ_UnlockWindow

NAME

WZ_UnlockWindow -- Fenstereingaben wieder erlauben

SYNOPSIS

```
result = WZ_UnlockWindow(winhandle)
```

D0 A0

```
ULONG WZ_UnlockWindow(struct WizardWindowHandle *);
```

FUNCTION

Benutzereingaben werden mit dieser Funktion wieder erlaubt.

WARNING

Ist das Fenster nicht gesperrt, dann kommt es zu schweren Systemfehlern. Wurde das Fenster mittels WZ_LockWindow() mehrmals gesperrt, dann muß es auch genauso oft mittels WZ_UnlockWindow wieder erlaubt werden. D.h. ein Aufruf muß nicht unbedingt tatsächlich Benutzereingaben wieder zulassen.

INPUTS

winhandle - WizardWindowHandler, dessen Fenster Eingaben wieder erlauben soll

RESULT

result - enthält, wie oft das Fenster noch mittels WZ_UnlockWindow() für Benutzereingaben zugänglich gemacht werden muß, dieser Wert ist gleich NULL wenn das Fenster wieder Benutzereingaben empfängt bei -1 ist ein schwerer Fehler aufgetreten.

SEE ALSO

WZ_LockWindow()

1.42 wizard.library/WZ_UnlockWindows

NAME

WZ_UnlockWindows -- Fenstereingaben wieder erlauben

SYNOPSIS

```
WZ_UnlockWindows(surface)
```

A0

```
VOID WZ_UnlockWindows(APTR);
```

FUNCTION

Diese Funktion führt für alle Fenster ein WZ_UnlockWindow() durch,

die zu einer Oberflächenbeschreibung gehören.

INPUTS

surface - Returnwert von WZ_OpenSurface()

SEE ALSO

WZ_LockWindows()
