Group.mui

Super class

Area.mui

Inherited by

Argstring.mui
Coloradjust.mui
Cycle.mui
Keyadjust.mui
Listview.mui
Mccprefs.mui

Palette.mui
Panel.mui
Popstring.mui
Radio.mui
Register.mui
Scrollbar.mui
Scrollgroup.mui
Title.mui
Virtgroup.mui

Background

Group class is responsible for the complete layout of a MUI window. A group may contain any number of child objects, maybe buttons, cycle gadgets or even other groups.

Some attributes of group class define how the children of a group are layouted. You can e.g. tell your group to place its children horizontally (in a row) or vertically (in a column). Since every MUI object knows about its minimum and maximum dimensions, group class has everything it needs to do that job.

More sophisticated layout is possible by assigning different weights to objects in a group or by making a group two-dimensional.

Beneath the layout issues, a group object passes attributes and methods through to all of its children. Thus, you can talk and listen to any child of a group by talking and listening to the group itself.

Attributes

Attribute	Version	ISG	Type
MUIA_Group_ActivePage	V5	ISG	LONG
MUIA_Group_Child	V4	I..	Object *
MUIA_Group_ChildCount	V20	..G	LONG
MUIA_Group_ChildList	V4	..G	struct List *
MUIA_Group_Columns	V4	IS.	LONG
MUIA_Group_Forward	V11	.S.	BOOL
MUIA_Group_ForwardDepth	V20	.S.	ULONG
MUIA_Group_Horiz	V4	I..	BOOL
MUIA_Group_HorizCenter	V20	ISG	LONG
MUIA_Group_HorizSpacing	V4	ISG	LONG
MUIA_Group_LayoutHook	V11	I..	struct Hook *
MUIA_Group_PageMode	V5	I..	BOOL
MUIA_Group_Rows	V4	IS.	LONG
MUIA_Group_SameHeight	V4	I..	BOOL
MUIA_Group_SameSize	V4	I..	BOOL
MUIA_Group_SameWidth	V4	I..	BOOL
MUIA_Group_Spacing	V4	IS.	LONG

MUIA_Group_VertCenter	V20	ISG	LONG
MUIA_Group_VertSpacing	V4	ISG	LONG

Methods

Method	Version
MUIM_Group_AddHead	V8
MUIM_Group_AddTail	V8
MUIM_Group_ExitChange	V11
MUIM_Group_InitChange	V11
MUIM_Group_MoveMember	V16
MUIM_Group_Remove	V8
MUIM_Group_Reorder	V21
MUIM_Group_Sort	V4

MUIA_Group_ActivePage

NAME

MUIA_Group_ActivePage — V5 [ISG], LONG, 0x80424199

SPECIAL INPUTS

- MUIV_Group_ActivePage_First
- MUIV_Group_ActivePage_Last
- MUIV_Group_ActivePage_Prev
- MUIV_Group_ActivePage_Next
- MUIV_Group_ActivePage_Advance

FUNCTION

Set (or get) the active page of a page group. Only this active page is displayed, all others are hidden.

The value may range from 0 (for the first child) to numchildren-1 (for the last child). Children are adressed in the order of creation:

```
PageGroup
  Child Page_0_Object
  Child Page_1_Object
  Child Page_2_Object
  Child Page_3_Object
  End
```

Note: You may **never** supply an incorrect page value!

SEE ALSO

MUIA_Group_PageMode

MUIA_Group_Child

NAME

MUIA_Group_Child — V4 [I..], `Object *`, 0x804226e6

FUNCTION

You supply a pointer to a previously created MUI object here. This object will be treated as child of the group, the group is responsible for positioning the object.

Of course you can specify any number of child objects, limited only by available memory.

Normally, the value for a MUIA_Group_Child tag is a direct call to another MUI_NewObject(), children are generated "on the fly".

When a group is disposed, all of its children will also get deleted. If you supply a NULL pointer as child, the group object will fail and previously dispose all valid children found in the taglist.

This behaviour makes it possible to generate a complete application within one single (but long) MUI_NewObject() call. Error checking is not necessary since every error, even if it occurs in a very deep nesting level, will cause the complete call to fail without leaving back any previously created object.

EXAMPLE

Please have a look at some of the supplied example programs.

SEE ALSO

MUIA_Group_Horiz

MUIA_Group_ChildCount

NAME

MUIA_Group_ChildCount — V20 [..G], `LONG`, 0x80420322

FUNCTION

Returns the number of group members of a group object.

SEE ALSO

MUIA_Group_Child

MUIA_Group_ChildList

NAME

MUIA_Group_ChildList — V4 [..G], `struct List *`, 0x80424748

FUNCTION

This attribute returns a pointer to a struct List in which a group manages its children.

The only thing you are allowed to do with this list is to traverse through the children. You **MUST** use intuition.library/NextObject() for this purpose!

Never add or remove member directly, use the OM_ADDMEMBER/OM_REMMEMBER methods instead!

SEE ALSO

MUIA_Group_Child

MUIA_Group_Columns

NAME

MUIA_Group_Columns — V4 [IS.], `LONG`, 0x8042f416

FUNCTION

Indicate number of columns in a two dimensional group. If you use this tag, the total number of children must be dividable by the number of columns.

The children will be positioned in a two dimensional array, e.g. allowing easy creation of button fields (maybe for calculator).

The children in your taglist are always read line by line.

When MUI layouts two-dimensional groups, it does actually two layout calculations, one for the rows and one the columns. Parameters like weights and dimensions are handled this way:

- the minimum width of a column/row is the maximum minimum width of all objects in this column/row.

- the maximum width of a column/row is the minimum maximum width of all objects in this column/row.
- the weight of a column/row is the sum of all objects in this column/row.

Actually, there is no difference if you use MUIA_Group_Columns or MUIA_Group_Rows.

EXAMPLE

```
GroupObject
  MUIA_Group_Columns
  MUIA_Group_Child label1
  MUIA_Group_Child string1
  MUIA_Group_Child label2
  MUIA_Group_Child string2
  MUIA_Group_Child label3
  MUIA_Group_Child string3

  End
```

SEE ALSO

MUIA_Group_Rows, MUIA_Group_Horiz

MUIA_Group_Forward

NAME

MUIA_Group_Forward — V11 [.S.], BOOL, 0x80421422

FUNCTION

Attribute controlling behaviour of a SetAttrs() function. When added before other attributes instructs the function to forward or not attribute change to all the children of a group instead of only group object itsef.

Defaults to TRUE.

EXAMPLE

```
SetAttrsgroup
  MUIA_Group_Forward FALSE
  MUIA_Disabled state
  TAG_DONE
```

SEE ALSO

MUIA_Group_ForwardDepth

MUIA_Group_ForwardDepth

NAME

MUIA_Group_ForwardDepth — V20 [.S.], ULONG, 0x80428488

FUNCTION

Attribute controlling behaviour of a SetAttrs() function. When added before other attributes instructs the function to forward the following attributes only up to the given depth within the group's hierarchy. Deeper nested objects will be skipped.

Defaults to infinite recursion.

EXAMPLE

```
SetAttrsgroup
  MUIA_Group_Forward TRUE
  MUIA_Group_ForwardDepth
  MUIA_Text_Contents
  TAG_DONE
```

SEE ALSO

MUIA_Group_Forward

MUIA_Group_Horiz

NAME

MUIA_Group_Horiz — V4 [I..], BOOL, 0x8042536b

FUNCTION

Boolean value to indicate whether the objects in this group shall be layouted horizontally or vertically.

Defaults to FALSE.

This is the easy way of telling your group how it has to look like. If you want two-dimensional groups, you have to use MUIA_Group_Columns or MUIA_Group_Rows.

EXAMPLE

```
GroupObject
  MUIA_Group_Horiz TRUE
  MUIA_Group_Child obj1
  MUIA_Group_Child obj2
  MUIA_Group_Child obj3
```

```
End
```

SEE ALSO

MUIA_Group_Columns, MUIA_Group_Rows, MUIA_Group_Child

MUIA_Group_HorizCenter

NAME

MUIA_Group_HorizCenter — V20 [ISG], LONG, 0x8042cc64

SPECIAL INPUTS

- MUIV_Group_HorizCenter_Left
- MUIV_Group_HorizCenter_Center
- MUIV_Group_HorizCenter_Right

FUNCTION

When a group layouts its children and a child is smaller than the actual space the group wants it to fill this attributes defines the alignment.

Defaults to MUIV_Group_HorizCenter_Center.

SEE ALSO

MUIA_Group_VertCenter

MUIA_Group_HorizSpacing

NAME

MUIA_Group_HorizSpacing — V4 [ISG], LONG, 0x8042c651

FUNCTION

Number of pixels to be inserted between horizontal elements of a group.

Please use this tag wisely, you will override the user's prefered default setting!

SEE ALSO

MUIA_Group_Spacing, MUIA_Group_VertSpacing

MUIA_Group_LayoutHook

NAME

MUIA_Group_LayoutHook — V11 [I..], `struct Hook *`, 0x8042c3b2

FUNCTION

Since version 11 of muimaster.library, you have the ability to customize the way objects are placed in a group. Altough MUI features a very powerful builtin layout algorithm which serves well for almost all GUI related purposes, it might sometimes become handy to override this with custom code.

Imagine you want to build a multimedia document viewer which contains text objects, bitmap objects and buttons. An easy way for doing this is to simply create a subclass of group class which contains all the documents elements as MUI objects and which specifies a custom layout hook for the parent group. This hook is then responsible for placing the objects within the bounds of the parent group.

As soon as you specify a MUIA_Group_LayoutHook, the builtin layout calculation is skipped and your hook is called whenever MUI needs some information. Register A2 will contain a pointer to the parent group object and register A1 will contain a pointer to a struct MUI_LayoutMsg. The lm_Type field of this structure determines which kind of action MUI wants you to perform (see below), the lm_Children field is a pointer to a list of your group's children. By traversing through list list with the intuition function NextObject(), you can retrieve the children of the group.

If lm_Type == MUILM_MINMAX, MUI wants you to calculate your group's minimum, maximum and default sizes. At this time, the children of your group have already been asked for their dimensions. This allows you to traverse through the list of children and do some calculations depending on their min/max sizes. Use the macros _minwidth(child), _maxwidth(child), _minheight(child), _maxheight(child) for this purpose. Place the result of your calculations in the structure lm_MinMax of the MUI_LayoutMsg and exit your hook with a return value of 0.

If lm_Type == MUILM_LAYOUT, MUI has already placed the group object somewhere in a window and now wants you to place the children of this group. You have to traverse through the child list and calculate positions and sizes for each child. Use the function MUI_Layout() to tell the child where it should appear. You have to make sure that you don't place child objects outside of the parent group and you should generally avoid overlapping objects. Return TRUE if all children are placed, return FALSE if you were for some reasons unable to place your children.

If your previous min/max calculations were correct, your algorithms should not have problems to place all the children in the rectangle defined by the parent group. Its size will never be smaller as your specified minimum dimensions and never be larger as your specified maximum dimensions.

If your group is a virtual group, the width and height your layout hook receives are as big as the **visible** part of the virtual group. In this case, you are allowed to position your objects outside of the visible part, i.e. you are not limited to keep your objects inside the given width and height. Place them where you wish and set lm_Layout.Width and lm_Layout.Height to the width and height you really need for your objects before exiting. The virtual width and height of your group will be adjusted accordingly.

EXAMPLE

see MUI demo program Layout.c

SEE ALSO

muimaster.library/MUI_Layout()

MUIA_Group_PageMode

NAME

MUIA_Group_PageMode — V5 [I..], BOOL, 0x80421a5f

FUNCTION

Settings this attribute to TRUE makes the current group a page group. Page groups always display only one of their children. Which one can be adjusted with the MUIA_Group_ActivePage attribute.

Imagine you have a preferences window with several different pages, e.g. the MUI preferences with object, frame, image, font, screen, keyboard and system prefs. Instead of one separate window for each group, you could put all pages into a page group and have a cycle gadget for page switching. This will make your program easier to use since the user don't have to handle a lot of windows. However, he will not be able to work with more than one page at the same time.

Sizes are calculated as follows:

- the minimum width/height of a page group is the maximum minimum width/height of all its children.
- the maximum width/height of a page group is the minimum maximum width/height of all its children.

When the maximum width/height of a child in a page group is smaller than the minimum width/height of the page group (since it contains another child with big minimum width/height), the child will be centered.

Page groups are not limited in depth, children of a page group may of course be other page groups.

If you want to have a gadget only visible under certain conditions, you could make a page group containing this gadget and an empty rectangle object.

If you want TAB cycling for the objects in a page group, simply include all objects in the cycle chain (as if they all were visible at the same time).

EXAMPLE

demo program "Pages.c"

SEE ALSO

MUIA_Group_ActivePage

MUIA_Group_Rows

NAME

MUIA_Group_Rows — V4 [IS.], LONG, 0x8042b68f

FUNCTION

Indicate number of rows in a two dimensional group. If you use this tag, the total number of children must be dividable by the number of rows.

The children will be positioned in a two dimensional array, e.g. allowing easy creation of button fields (maybe for calculator).

The children in your taglist are always read line by line.

When MUI layouts two-dimensional groups, it does actually two layout calculations, one for the rows and one the columns. Parameters like weights and dimensions are handled this way:

- the minimum width of a column/row is the maximum minimum width of all objects in this column/row.
- the maximum width of a column/row is the minimum maximum width of all objects in this column/row.
- the weight of a column/row is the sum of all objects in this column/row.

Actually, there is no difference if you use MUIA_Group_Columns or MUIA_Group_Rows.

SEE ALSO

MUIA_Group_Columns, MUIA_Group_Horiz

MUIA_Group_SameHeight

NAME

MUIA_Group_SameHeight — V4 [I..], BOOL, 0x8042037e

FUNCTION

Boolean value to indicate that all children of this group shall have the same height. The exception are objects with an explicit fixed height, i.e. VSpace(x) like objects. This makes it possible to, for example, have a fixed size spacing in a column of objects with a dynamical but yet equal height.

BUGS

Up to version 5 of groupclass, using MUIA_Group_SameHeight could make objects larger than their maximum height. This has been fixed for version 6.

SEE ALSO

MUIA_Group_SameSize, MUIA_Group_SameWidth

MUIA_Group_SameSize

NAME

MUIA_Group_SameSize — V4 [I..], BOOL, 0x80420860

FUNCTION

This is a shorthand for MUIA_Group_SameWidth and MUIA_Group_SameHeight, it sets both of these attributes at once.

Using MUIA_Group_SameSize, you don't need to think if your group is horizontal or vertical, both cases are handled automatically.

Forcing all objects of a group to be the same size is e.g. useful for a row of buttons. It is visually more attractive when these buttons have equal sizes instead of being just as big as the text within.

Note that objects with an explicitly set fixed width or height are excluded from this rule. This makes it possible to, for example, have a fixed size spacing in a row of equally sized buttons.

BUGS

Up to version 5 of groupclass, using MUIA_Group_SameSize could make objects larger than their maximum size. This has been fixed for version 6.

EXAMPLE

```
GroupObject
  MUIA_Group_Horiz TRUE
  MUIA_Group_SameSize TRUE
  MUIA_Group_Child but1
  MUIA_Group_Child but2
  MUIA_Group_Child but3
```

```
End
```

SEE ALSO

MUIA_Group_SameWidth, MUIA_Group_SameHeight

MUIA_Group_SameWidth

NAME

MUIA_Group_SameWidth — V4 [I..], BOOL, 0x8042b3ec

FUNCTION

Boolean value to indicate that all children of this group shall have the same width. The exception are objects with an explicit fixed width, i.e. HSpace(x) like objects. This makes it possible to, for example, have a fixed size spacing in a row of equally sized buttons.

BUGS

Up to version 5 of groupclass, using MUIA_Group_SameWidth could make objects larger than their maximum width. This has been fixed for version 6.

SEE ALSO

MUIA_Group_SameSize, MUIA_Group_SameHeight

MUIA_Group_Spacing

NAME

MUIA_Group_Spacing — V4 [IS.], LONG, 0x8042866d

SPECIAL INPUTS

- MUIV_Group_Spacing_Default
- MUIV_Group_Spacing_Percent(p)

FUNCTION

This is a shorthand for MUIA_Group_HorizSpacing and MUIA_Group_VertSpacing, it sets both of these attributes at once.

Using MUIA_Group_Spacing, you don't need to think if your group is horizontal or vertical, both cases are handled automatically.

Note that setting a spacing value for a group overrides the user's default settings. Please use it only if you have a good reason.

EXAMPLE

```
GroupObject
  MUIA_Group_Horiz TRUE
  MUIA_Group_Spacing
  MUIA_Group_Child obj1
  MUIA_Group_Child obj2
  End
```

SEE ALSO

MUIA_Group_HorizSpacing, MUIA_Group_VertSpacing

MUIA_Group_VertCenter

NAME

MUIA_Group_VertCenter — V20 [ISG], LONG, 0x8042c008

SPECIAL INPUTS

- MUIV_Group_VertCenter_Top
- MUIV_Group_VertCenter_Center
- MUIV_Group_VertCenter_Bottom

FUNCTION

When a group layouts its children and a child is smaller than the actual space the group wants it to fill this attributes defines the alignment.

Defaults to MUIV_Group_VertCenter_Center.

SEE ALSO

MUIA_Group_HorizCenter

MUIA_Group_VertSpacing

NAME

MUIA_Group_VertSpacing — V4 [ISG], LONG, 0x8042e1bf

FUNCTION

Number of pixels to be inserted between vertical elements of a group.

Please use this tag wisely, you will override the user's prefered default setting!

SEE ALSO

MUIA_Group_Spacing, MUIA_Group_HorizSpacing

MUIM_Group_AddHead

NAME

MUIM_Group_AddHead — V8, 0x8042e200

SYNOPSIS

```
DoMethod(obj, MUIM_Group_AddHead, Object *obj);
```

FUNCTION

This method adds an object at the head of the group's object list.

INPUTS

Object *obj
: Pointer to an object to be added.

RESULT

FALSE for failure, TRUE for success.

SEE ALSO

MUIM_Group_AddTail, MUIM_Group_InitChange, MUIM_Group_ExitChange

MUIM_Group_AddTail

NAME

MUIM_Group_AddTail — V8, 0x8042d752

SYNOPSIS

```
DoMethod(obj, MUIM_Group_AddTail, Object *obj);
```

FUNCTION

This method adds an object at the tail of the group's object list.

INPUTS

Object *obj
: Pointer to an object to be added.

RESULT

FALSE for failure, TRUE for success.

SEE ALSO

MUIM_Group_AddHead, MUIM_Group_InitChange, MUIM_Group_ExitChange

MUIM_Group_ExitChange

NAME

MUIM_Group_ExitChange — V11, 0x8042d1cc

SYNOPSIS

```
DoMethod(obj, MUIM_Group_ExitChange);
```

FUNCTION

Terminates MUIM_Group_InitChange state.

SEE ALSO

MUIM_Group_InitChange

MUIM_Group_InitChange

NAME

MUIM_Group_InitChange — V11, 0x80420887

SYNOPSIS

```
DoMethod(obj, MUIM_Group_InitChange);
```

FUNCTION

Prepares a group for dynamic adding/removing of objects. MUI 3 offers the possibility to dynamically add/remove children from groups, even when the window that contains these objects is currently open. To be able to do this, you must first put the group into a special "exchange" state by using this method. Then you can add/remove children at will. When you're done, use MUIM_Group_ExitChange to make MUI recalculate the display.

RESULT

Returns NULL on failure.

EXAMPLE

```
DoMethodgroup MUIM_Group_InitChange

  DoMethodgroup OM_REMMEMBER somechild
  DoMethodgroup OM_REMMEMBER somechild2

  DoMethodgroup OM_ADDMEMBER somenewchild

  DoMethodgroup MUIM_Group_ExitChange
```

SEE ALSO

MUIM_Group_ExitChange

MUIM_Group_MoveMember

NAME

MUIM_Group_MoveMember — V16, 0x8042ff4e

SYNOPSIS

```
DoMethod(obj, MUIM_Group_MoveMember, Object *o, LONG pos);
```

FUNCTION

This method rearranges one child in a group. The child is removed first and then inserted according to the second parameter.

INPUTS

Object *o
: object to rearrange, must be child of the group.

LONG pos
: new position for the child object:
  - ◊ pos == 0: insert as first child of the group.
  - ◊ pos == -1: insert as last child of the group.
  - ◊ pos > 0: insert after group member <pos>, counting from the beginning of the group.
  - ◊ pos < -1: insert after group member pos>, counting from the end of the group.

RESULT

The result value is currently undefined.

NOTES

Always enclose in MUIM_Group_InitChange and MUIM_Group_ExitChange.

EXAMPLE

```
hgr  HGroup
  Child o1
  Child o2
  Child o3
  Child o4
  Child o5
  End

DoMethodhgr MUIM_Group_InitChange

  DoMethodhgr MUIM_Group_MoveMember o1
  DoMethodhgr MUIM_Group_ExitChange
```

SEE ALSO

MUIM_Group_Sort, MUIM_Group_InitChange, MUIM_Group_ExitChange

MUIM_Group_Remove

NAME

MUIM_Group_Remove — V8, 0x8042f8a9

SYNOPSIS

```
DoMethod(obj, MUIM_Group_Remove, Object *obj);
```

FUNCTION

This method removes an object from the group's object list. This method is the same as OM_REMMEMBER.

INPUTS

Object *obj
: Pointer to an object to be removed.

RESULT

FALSE for failure, TRUE for success.

SEE ALSO

MUIM_Group_AddHead, MUIM_Group_AddTail, MUIM_Group_InitChange, MUIM_Group_ExitChange

MUIM_Group_Reorder

NAME

MUIM_Group_Reorder — V21, 0x80426c3f

SYNOPSIS

```
DoMethod(obj, MUIM_Group_Reorder, Object *after, Object *array[1]);
```

FUNCTION

Reorder the children of a group.

INPUTS

Object *after
: the object to reorder the children after. Special values are
  - ◊ 0: reorder the objects at the front of the Group object's child list.
  - ◊ -1: reorder the objects at the end of the Group object's child list.

Object *array[1]
: a NULL terminated array of objects to be reordered.

EXAMPLE

```
DoMethodgroup MUIM_Group_Reorder  child1 child2 child3
```

SEE ALSO

MUIM_Group_MoveMember, MUIM_Group_Sort

MUIM_Group_Sort

NAME

MUIM_Group_Sort — V4, 0x80427417

SYNOPSIS

```
DoMethod(obj, MUIM_Group_Sort, Object *obj[1]);
```

FUNCTION

This method rearranges the order of the children stored in a group object.

INPUTS

Object *obj[1]
: array that contains **all** the children of the group in the desired order. The array must be terminated with a NULL entry.

RESULT

The result value is currently undefined.

NOTES

You must always pass all objects of your group!

EXAMPLE

```
hgr  HGroup
  Child o1
  Child o2
  Child o3
  Child o4
  End

DoMethodhgr MUIM_Group_Sort o4 o3 o2 o1
```

SEE ALSO

MUIM_Group_MoveMember, MUIM_Group_InitChange, MUIM_Group_ExitChange

---