

Toolbar

Benny Kjær <benny@diku.dk>

COLLABORATORS

	<i>TITLE :</i> Toolbar		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY	Benny Kjær <benny@diku.dk>	January 19, 2025	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1	Toolbar	1
1.1	Toolbar - AmigaGuide documentation	1
1.2	Introduction	1
1.3	Features	2
1.4	How to install	3
1.5	Usage	3
1.6	Toolbar History	4
1.7	InfoText History	6
1.8	Frequently Asked Questions	6
1.9	To Do	7
1.10	Magic User Interface	7
1.11	Author	8
1.12	Credits	8
1.13	PolymorphWare	8

Chapter 1

Toolbar

1.1 Toolbar - AmigaGuide documentation

The Toolbar class for MUI

Toolbar.mcc Version 15.5 © 1997-98

Toolbar.mcp Version 15.4 © 1997-98

InfoText.mcc Version 15.3 © 1997-98

Programmed by

Benny Kjær Nielsen

Table of contents

Introduction.....	What is it?
Features.....	What can it do?
Install.....	How to install?
Usage.....	How do I use it?
Toolbar History.....	What has been changed?
InfoText History.....	What has been changed?
FAQ.....	What/Where/Why/Who?
ToDo.....	What is going to change?
MUI.....	What is MUI?
Author.....	Who am I?
Credits.....	Who made this possible?
PolymorphWare.....	PolyWhat?

1.2 Introduction

Introduction

The toolbar custom class supplies an easy way of implementing a toolbar in your programs. It features a multitude of different settings and the

majority of these are configurable for the user via the MUI preferences system.

To use `Toolbar.mcc,p` you will naturally need to have MUI installed on your machine. Furthermore you will need to have the `datatypes.library`. If you want to run the demo program then you will also need to install `InfoText.mcc` which is also present in this archive.

1.3 Features

`Toolbar.mcc,p`

The easiest way to get an impression of the toolbar class is to read the autodoc and to test the demo program. Don't forget trying out all the settings possible via `Toolbar.mcp`. The most important features are:

- handles three types of images: normal, selected and ghosted
- automatically adapts to image-sizes
- horizontal/vertical appearance
- mutual exclusion
- powerful internal notification-system
- a multitude of user-configurable settings
- ...

The toolbar also features an image cache to limit the need for loading and remapping identical image files. The condition for reusing a remapped image is that the filename and the screen must be the same. If all windows using a particular image file are disposed then the image will be disposed as well. I might change this behaviour to keeping images after windows are closed or programs are quitted, but it will certainly be a prefs-option since you could end up with a bundle of unused images in the cache.

`InfoText.mcc`

This is simply a subclass of `Text.mui` which features a fancy border. This is especially useful for a class such as `Toolbar.mcc` (see the example code in the `Toolbar.mcc` distribution archive), but can also be used for other purposes.

It is VERY important to notice that the class is a subclass of `Text.mui`.

This means that all the attributes described in the autodoc "`MUI_Text.doc`" can be used with this class as well.

Some extra features of the `InfoText` class is timeout texts and fall back texts - see the autodoc for details.

1.4 How to install

How to install

You can easily install these classes manually.

Simply copy the files in 'Libs/MUI' to the drawer 'MUI:Libs/MUI'

If you are a developer and want to use the toolbar and/or the infotext class then you also need to copy the files in Developer/C/Include/mui/ to your own include directory.

Some catalog files are also available in the 'Catalogs/' directory. If you want to use these then copy them to 'MUI:Locale/Catalogs/'

1.5 Usage

Usage

Normal user

As a user you should have no trouble with these classes. They are not complicated in any way. Simply try all the settings in the preferences-program and observe the effect. If you experience any problems you can always send me an [email](#) if you want.

Creative user

If you wish to change the images used in a toolbar then you should go hunting for the files containing images. You will find 1, 2 or 3 imagefiles which will correspond to normal, ghosted and selected images. These can be changed and the toolbar will automatically (when reloaded) adapt to the new situation. You just have to follow these rules when creating the images.

- every file must contain the correct number of images (set up horizontally)
- every image must have the same width and the same height
- all images must be separated with a line
- you must save the images as a brush with the exact width and height
- the images in all three files (you don't have to make them all) must have the same width and height

If the total width of the images in a file doesn't make sense compared to the number of buttons specified by the programmer then a warning will be displayed when the toolbar is used.

NOTICE: You do not have to use the same width and/or height as in the images supplied with the program. The toolbar automatically adapts to a different width/height. Just make sure you obey the rules mentioned above.

Developer

If you are a developer and want to use one of these classes then read the autodocs and take a look at the files in 'Developer/C/' - even if you are not a C-programmer.

There are a few points that you should take especially care with.

- ALWAYS specify all three types of images (filenames), so that users can add images you didn't make yourself. Write the filenames somewhere in your documentation.
- Do NOT use MUIA_Font if you do not really want to override the users settings
- Make sure that you save the images with the correct background colour (transparency colour). See the [FAQ](#)-section for more info.
- ...

1.6 Toolbar History

History of Toolbar.mcc

Version 15.5 (20-Dec-98)

- changed the behaviour when no images was found - now it simply uses the TextOnly-mode
- to accomodate problems with hotkeys the toolbar will only react on 'ramiga+hotkey' if there is an active object in the window.
- fixed offset-problems with background images
- fixed enforcer-hits in MUIM_Show
- corrected bug concerning registerpages (e.g. in YAM's read-window)
- corrected problem with pressing button simultaneously with setting it to be ghosted
- corrected the behaviour of the toolbar when setting MUIA_Disable
- NEW: MUIA_Toolbar_Reusable
- NEW: MUIM_Toolbar_ReloadImages
- NEW: image-cache (see [features](#))
- improved the verification of image-sizes
- updated/extended the autodoc and the guide

Version 15.4 (9-Feb-98)

- corrected the behaviour of MUIA_Toolbar_ParseUnderscore
 - some additions/changes in the autodoc
 - NEW: button-spacing - configurable spacing between buttons
 - NEW: text-only mode (no images, just the labels)
 - NEW: more notify-triggers: MUIV_Toolbar_Notify_LeftEdge/RightEdge/TopEdge/BottomEdge useful for easily placing a popup-menu.
-

- NEW: 4 placement-attributes: MUIA_Toolbar_LeftEdge/RightEdge/TopEdge/BottomEdge
also useful for placing a popup-menu.

- NEW: more imagetypes supported, MUIV_Toolbar_ImageType_Object and _Memory

Version 15.3 (25-Nov-97)

- removed a bug in OM_NEW (redundant code that could cause enforcer hits)

Version 15.2 (23-Nov-97)

- removed a lot of obsolete code from demo program and added some symbolic names

- enabled compiler optimizations (reduced size of Toolbar.mcc with 4k)

- NEW: MUIM_Toolbar_MultiSet

- added support for _ in the tooltexts - see MUIA_Toolbar_ParseUnderscore

- correct remapping and use of background colour (bmh_Transparent)

- uses picture datatypes v43 - does it work on a graphics card?

- removed some enforcer-hits and related errors

Version 15.1 (5-Oct-97)

- improved hotkey-handling

- changed the names of the descriptionflags

- BUG: choice of font could give serious errors

- BUG: hotkeys didn't work

- BUG: mutual-exclude didn't work

- the serial-number is now correct

- wrote most of the autodoc

- implemented MUIV_Notify_Self/Window/Application

- implemented MUIV_EveryTime/MUIV_TriggerValue

- NEW: TDT_IGNORE

- removed TDT_TOGGLE

- changed descriptionflags NEW: TDF_RADIO / TDF_TOGGLE

- hotkey underlined

- added KillNotifyObj

History of Toolbar.mcp

Version 15.4 (20-Dec-98)

- rearranged some of the elements

- changed the about-text

- removed the 'dimmed'-option because it was never implemented :)

- removed the 'inside/outside' option because it has not yet been
implemented

- NEW: 'Use ghosted/selected images if available'

Version 15.3 (9-Feb-98)

- NEW: localization - currently the only translation is Danish

- completely changed design

- NEW: text-only / remap precision (OBP_PRECISION) / ghosttype / toolspace

Version 15.2 (23-Nov-97)

- new prefs-image

Version 15.1 (5-Oct-97)

- NEW: prefsoption which let you enable/disable the tooltexts
- NEW: two prefsoptions called InnerSpaceText/NoText

1.7 InfoText History

History of InfoText.mcc

Version 15.3 (9-Feb-98)

- NEW: MUIA_InfoText_FallBackText [ISG] - see autodoc for details
- MUIA_InfoTextContents is now [.SG]
- MUIA_InfoTextExpirationPeriod is now [ISG]
- some remapping problems finally solved (I hope)
- removed bug with simple refresh windows
- changes/additions in autodoc

Version 15.2 (25-Nov-97)

- removed a bug in MUIM_Setup (used the non-valid _window(obj))
- bumped the version number correctly

Version 15.1 (23-Nov-97)

- also remapping correctly
- NEW: MUIA_InfoText_Contents - this special attribute has a timeout feature
- NEW: MUIM_InfoText_TimeOut - force the text to 'expire'
- NEW: MUIA_InfoText_ExpirationPeriod

1.8 Frequently Asked Questions

Frequently Asked Questions

Q: I have a thick border around the tool icons which is not in the same colour as the background of the toolbar picture. Why is this colour not correct?

A: The toolbar class uses the datatypes library to handle the pictures.

This includes loading, remapping and picture information such as size and transparency colour. The background colour is based on the transparency colour, but unfortunately some datatypes are programmed incorrectly and does not supply the correct transparency colour. This is e.g. the case with the ilbm datatype in the archive PictDT43-760.lzx. Another example is 'akPNG.datatype'. You can get a much better PNG datatype from Cloanto

which works correctly.

To solve the problem you could convert the picture to a format that does handle the transparency colour correctly.

--

Q: What is the transparency colour?

A: The transparency colour is best described as the background colour of a picture. In most paint programs you can use the right mouse button to select a colour to be the transparency colour.

--

1.9 To Do

The future of the Toolbar class

- improve notify - MUIV_Toolbar_Notify_Active
- new ghost-routine
- tab-support
- use MUI's frames
- noborder option (like in Win95-Word)
- text outside buttons
- whatever YOU suggest...

1.10 Magic User Interface

This application uses

MUI - MagicUserInterface

(c) Copyright 1993-97 by Stefan Stuntz

MUI is a system to generate and maintain graphical user interfaces. With the aid of a preferences program, the user of an application has the ability to customize the outfit according to his personal taste.

MUI is distributed as shareware. To obtain a complete package containing lots of examples and more information about registration please look for a file called "muiXXusr.lha" (XX means the latest version number) on your local bulletin boards or on public domain disks.

If you want to register directly, feel free to send
DM 30.- or US\$ 20.-

to

Stefan Stuntz

Eduard-Spranger-Straße 7

80935 München

GERMANY

Support and online registration is available at

<http://www.sasg.com/>

1.11 Author

How to contact me

If you have any ideas, critical remarks or bugs to report you can contact me on the following electronic address:

floyd@amiga.dk

Should this address not work then you can try the following:

benny@diku.dk

You could also visit my homepage which will always contain the latest revision.

<http://www.diku.dk/students/benny/>

If you want to write me a genuine old fashioned postcard then please don't hesitate. Write me at:

Benny Kjær Nielsen

Karensgade 8, st.th.

2500 Valby

Denmark

1.12 Credits

Credits

In the development of these classes I could not have been without the help of Allan Odgaard. His suggestions and ideas (and even code) has influenced most of the basic design. If you don't like the toolbar - blame him :-)

If you like the toolbar you will probably also like a lot of Allan's programs and classes - they can be found at: <http://www.diku.dk/students/duff/>

Also thanks to the (active) beta testers:

Allan Odgaard - <http://www.diku.dk/students/duff/>

Ole Friis

Marcel Beck - <http://www.yam.ch/>

Szymon Pura - <http://friko.onet.pl/cz/szymp>

Massimiliano Origgi - <http://lasvegas.pointtest.com/intui/>

Mario Cattaneo - <http://fsinfo.cs.uni-sb.de/~cattaneo/>

1.13 PolymorphWare

PolymorphWare

Toolbar.mcc/mcp is released under the concept of Polymorphware. This means that whatever you demand of your users I will demand from you. A couple of examples

might make it clearer. If your software uses my class and you have released it as postcard-ware then you have to mail me a postcard. If you have released it as shareware then you have to pay me the shareware-fee.

There is one exception though. Commercial software-developers will have to contact me to get an explicit permission to use my class.

This concept was first used by Allan Odgaard :)

InfoText.mcc is freeware. You can use it in whatever you like even commercial programs.