SpeedBar-English

Alfonso Ranieri

COLLABORATORS			
	TITLE : SpeedBar-English		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Alfonso Ranieri	August 24, 2022	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

SpeedBar-English

1.1 SpeedBar Programmers Guide

```
                SpeedBar User Guide
Copyright © 2000-2002 Alfonso Ranieri and Simone Tellini
```


1.2 SpeedBar - Introduction

```
Introduction

 SpeedBar allows you to easily create toolbars similar to those
 you can see in Windows programs (although you can get a
 standard look as well ;)
```

1.3 SpeedBar - Author

```
Author

  SpeedBar was originally created by Simone Tellini
  <wiz@vapor.com>.

  Since version 12.0 the project is manteined and developed by
  Alfonso Ranieri <alforan@tin.i>.

  The support page is located at
```

```
  http://digilander.libero.it/asoft/
```

1.4 SpeedBar - Warning, Requirements, Installation and Distribution

```
Warning, Requirements, Installation and Distribution

  Warning
  THIS SOFTWARE AND INFORMATION ARE PROVIDED AS IS. ALL USE IS
  AT YOUR OWN RISK, AND NO LIABILITY OR RESPONSIBILITY IS
  ASSUMED. NO WARRANTIES ARE MADE.

  Requirements
  The library needs AmigaOS, version 3 or higher.

  Installation
  Use the installation script.

  Distribution
  These classes are free for users: they don't have to pay
  anything; they are not free for developers: I want to receive
  a registered copy of your program if it uses them.
```

1.5 SpeedBar - Programmers

```
Programmers

 SpeedBar is very easy to program. All you have to do is to
 supply a description of the buttons and a description of the
 images to use.

 The buttons are defined via MUIA_SpeedBar_Buttons tag. You
 have to pass a struct MUIS_SpeedBar_Button array, the fields
 of which are:

 - Img
   the index of the image to use for this button

 - Text
   the label for this button

 - Help
   an help string to use as ShortHelp

 - Flags
   flags for this button

 - Class
   if you subclassed SpeedButton.mcc, pass your subclass here.

 - Object
   this will be filled after the object was created. Just don't
   use it :-)
```

```
Example:

struct MUIS_SpeedBar_Button buttons[] =
{
    {0, "_Get", "Get the disc.", 0, NULL},
    {1, "Sa_ve", "Save the disc.", 0, NULL},
    {2, "_Stop", "Stop the connection.", 0, NULL},
    {MUIV_SpeedBar_Spacer},
    {3, "_Disc", "Disc page.", 0, NULL},
    {4, "_Matches", "Matches page.", 0, NULL},
    {5, "_Edit", "Edit page.", 0, NULL},
    {MUIV_SpeedBar_End},
};

Note that the array must be always terminated with
MUIV_SpeedBar_End.

You may add other buttons at any time via
MUIM_SpeedBar_AddButton and MUIM_SpeedBar_AddSpacer .

The button definition is copied, so you may discard it after
use. Help is not copied !

The description of the image may be supplied in different
ways:

MUIA_SpeedBar_Images
you load the single images and set up an array of struct
MyBrush * . The array may be freed after use (the array, not
the BitMaps!).

MUIA_SpeedBar_Pics
you want SpeedBar.mcc to load the single images for you. Pass
here a NULL-terminated array of STRPTR, names of the images to
load via datatypes. The array may be freed after use.

MUIA_SpeedBar_Strip
the images are saved as a Strip: all together separeted via a
single blank pixels column. The strip will be load via
datatypes. The number of the buttons is derived via
MUIA_SpeedBar_Buttons or as defined in
MUIA_SpeedBar_StripButtons (if you want to add buttons later).
Pass a STRPTR, the name of the strip, here. It may be freed
after use.

MUIA_SpeedBar_StripBrush
as the above but you load the strip by yourself. Pass a struct
MyBrush *, definition of the strip, here. It may be freed
after use (the pointer not the BitMap!).

When using a strip, all the pics must have the same width. No
requester is shown and no error is reported if the supplied
strip is invalid. You will see that by yourself :-)

The MUIA_SpeedBar_PicDrawer attribute me be used to define a
path for the pics and the strip to be be load via datatypes.
```

Never cache the single buttons objects pointer: if you set the attributes:

- MUIA_Group_Horiz
- MUIA_SpeedBar_BarSpacer

and if the object already setup:
- MUIA_SpeedBar_Borderless
- MUIA_SpeedBar_ViewMode
- MUIA_SpeedBar_SmallImages
- MUIA_SpeedBar_Sunny

all the buttons are disposed an recreated!

To invoke a method over a buttons use MUIM_SpeedBar_DoOnButton or MUIM_SpeedBar_GetObject .

Always let the user modify the aspect of the bar via:

- MUIA_SpeedBar_BarSpacer
- MUIA_SpeedBar_Borderless
- MUIA_SpeedBar_ViewMode
- MUIA_SpeedBar_SmallImages
- MUIA_SpeedBar_Sunny
- MUIA_SpeedBar_Raising
- MUIA_SpeedBar_LabelPosition
- MUIA_SpeedBar_EnableUnderscore

For subclass: set the above values before calling the super method in you MUIM_Setup: this will avoid to recrate all the buttons.

If possible, put the bar in a Virtgroup: this will not stress the user if very big fonts and or images are used.

MUIA_SpeedBar_EnableUnderscore is a S attribute, but it must be inited as TRUE to be settable.

If the action taken at a button pressing takes very long time, consider to PushMethod it rather than notify; that means sometimes

```
DoMethod(DoMethod(sb,MUIM_SpeeBar_GetButton,I),MUIM_Notify,
    MUIA_Pressed,FALSE,app,X,MUIM_Application_PushMethod,
    obj,Y,...);
```

may be better than

```
DoMethod(DoMethod(sb,MUIM_SpeeBar_GetButton,I),MUIM_Notify,
    MUIA_Pressed,FALSE,obj,Y,...);
```

Note the BitMaps passes in the BitMap fiels of struct MyBrush have to be standard V42 BitMaps; while remapping they are converted to screen friend ones.