buttonbase

COLLABORATORS			
	TITLE : buttonbase		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		December 6, 2024	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

buttonbase

1.1 Buttonbase Plugin and its Subclasses

```
Documentation for:  buttonbase.m
                    newbutton.m
                    newimagebutton.m

by Victor Ducedre <victord@netrover.com>

    Brief Summary
    About buttonclass.m

 newbutton.m                          newimagebutton.m

    Usage                             Usage
    Methods                           Methods
    Attributes                        Attributes
    Exceptions                        Exceptions

    History
    Author/Copyright/Thanks
```

1.2 Contacting the author

```
Copyright information

All files in this distibution are Copyright © 1998 Victor Ducedre, except
where otherwise indicated.

You are free to use these files in your own programs, and you may modify
the supplied sources for your own purposes, but you may not redistribute
without at least being so courteous as to ask me first; the sole exception
is the EasyPLUGINs collection.

Contacting the Author

Feedback is always welcome, especially where bugs are concerned.  I can be
reached via Email at <victord@netrover.com>, although I'm not always
```

proud of my response time.

Thanks to:

Wouter and Jason, for the wonderfulness that is EasyGUI, and

Ali Graham, for organizing and maintaining the EasyPLUGINs collection;

1.3 Brief Summary

This set of EasyGUI Plugins is an interface for my ButtonClass BOOPSI gadget, designed as a replacement for the 'button.library' Plugins that come with EasyGUI.

newbutton.m is a standard text button that supports keyboard shortcuts.

newimagebutton.m is an image button that supports normal and selected images, and can be set either to the size of your image or to dimensions of your choosing.

buttonbase.m? This is a Plugin of which the other two are subclasses, containing all the common elements of newbutton.m and newimagebutton.m. You don't use it directly in your EasyGUI gadget list.

But why use this instead of 'button.library'? While it does add to your final code size by not using an external library, it does eliminate any confusion between Commodore's 'button.library' and ClassAct's included 'button.library', which I've found is not as compatible with C='s as the ClassAct folks would have us believe. And besides, this does add more features, like those mentioned above, and is actually Style Guide compliant. (I wonder if Commodore actually ever read their own Style Guide... :-)

1.4 History

5-Jun-98 First Release (version 1.0)

1.5 newbutton.m/Usage

Notes about Usage

* To install the modules, copy the contents of the 'plugins' and 'gadgets' directories to their respective directories in Emodules:

* To create a new newbutton object, use, e.g.:

```
MODULE 'gadgets/buttonclass',
       'plugins/buttonbase', 'plugins/newbutton'
[...]
```

```
        DEF btn:PTR TO newbutton
        NEW btn.button([..., TAG_DONE])
```

*   The constant NEWBUTTON (which =PLUGIN) is available to be used in your EasyGUI gadget list, as in:

```
        [NEWBUTTON, {actionfunction}, nb]
```

This can make plugins in gadget lists easier to identify, especially if you use a lot of different plugins.

<<   Start
Methods   >>

1.6 newbutton.m/Methods

Methods

button(tags)

    The contructor for this method, where tags is a PTR TO a list of tag items from those attributes marked [I]. Most values default to something reasonable, so you can simply use [NB_TEXT, 'Text', TAG_DONE] as your list.
    You must open utility.library prior to calling this function, or it will raise a "util" exception.
    This method has no return value.

set(attr, val)

    Sets any attribute to the value given, where attr is any attribute that is marked [S], and val is some reasonable value for that attribute.
    This method has no return value.

value, check:=get(attr)

    Gets the value for the specified attribute, where attr is any attribute marked [G]. The second return value, check, is TRUE if attr is in fact "gettable", otherwise FALSE (be sure to check for this).

setcolour(colour, val)

    This is a (probably temporary) extra function provided to allow you to change the colours used to render the button, where colour is one of BUT_TEXTPEN, BUT_FILLPEN, BUT_FILLTEXTPEN, BUT_BACKGROUNDPEN (defined in buttonclass.m), and val is a reasonable number for that pen (no checking is done)

END must be called for each NEWed object.

<<   Usage
Attributes >>

1.7 newbutton.m/Attributes

```
Attributes

    In addition to the common attributes in buttonbase, the newbutton
Plugin also supports:

NB_TEXT                                 [IS.]

    The label which is to appear in the button.  Including a "_" in this
string will automatically assign the next character as the keyboard
shortcut, and will indicate it as such in the label. (Default: No label
text)
    CAUTION!!!  When set()ing this attibute, no adjustment is made to the
size of the gadget (since I don't know how to signal EasyGUI to resize
itself).  Supplying a new label that's larger than the one set in the
constructor HAS NOT BEEN TESTED, and may produce undesireable effects!


<<    Methods
Exceptions >>
```

1.8 newimagebutton.m/Usage

```
Notes about Usage

*   To install the modules, copy the contents of the 'plugins' and
'gadgets' directories to their respective directories in Emodules:

*   To create a new newimagebutton object, use, e.g.:

        MODULE 'gadgets/buttonclass',
               'plugins/buttonbase', 'plugins/newimagebutton'
        [...]
        DEF ibtn:PTR TO newimagebutton
        NEW ibtn.button([..., TAG_DONE])

*   The constant NEWIMAGEBUTTON (which =PLUGIN) is available to be used in
your EasyGUI gadget list, as in:

        [NEWIMAGEBUTTON, {actionfunction}, nb]

This can make plugins in gadget lists easier to identify, especially if
you use a lot of different plugins.

<<      Start
Methods    >>
```

1.9 newimagebutton.m/Methods

```
Methods
```

```
button(tags)

    The contructor for this method, where tags is a PTR TO a list of tag
items from those attributes marked [I].  Most values default to something
reasonable, so you can simply use [NIB_IMAGE, img, TAG_DONE] as your list.
    You must open utility.library prior to calling this function, or it
will raise a "util" exception.
    This method has no return value.

set(attr, val)

    Sets any attribute to the value given, where attr is any attribute
that is marked [S], and val is some reasonable value for that attribute.
    This method has no return value.

value, check:=get(attr)

    Gets the value for the specified attribute, where attr is any attribute
marked [G].  The second return value, check, is TRUE if attr is in fact
"gettable", otherwise FALSE (be sure to check for this).

END must be called for each NEWed object.

<<      Usage
Attributes >>
```

1.10 newimagebutton.m/Attributes

```
Attributes

    In addition to the common attributes in buttonbase, the newimagebutton
Plugin also supports:

NIB_IMAGE                               [IS.]

    The image (must be in chip mem; use tools/copylist) which is to appear
centred in the button.  This attribute must be provided, or "nbut" will be
raised.
    Note that when changing the image with set(), the new image is only
accepted and changed if the width and height are <= those of the original
image.  The gadget will not adjust its size to fit.

NIB_SELECTIMAGE                         [IS.]

    This is an optional second image, which will be rendered when the
button is selected.  Ideally, this image should be the same dimensions as
the image provided with NIB_IMAGE, or you should allow for this using
NIB_WIDTH and NIB_HEIGHT
    Note that when changing the image with set(), the new image is only
accepted and changed if the width and height are <= those of the original
image.  The gadget will not adjust its size to fit. (Default: NIL)

NIB_WIDTH                               [I.G]

    The width of the button.  It will adjust to the width of the image if
```

you have provided too small a value or if you omit this tag. (Default: width of NIB_IMAGE image)

NIB_HEIGHT [I.G]

The height of the button. It will adjust to the height of the image if you have provided too small a value or if you omit this tag. (Default: height of NIB_IMAGE image)

<< Methods
Exceptions >>

1.11 Exceptions

Exceptions

"util" will be raised by buttonbase/button() if utility.library isn't open.
"nbut" will be raised by buttonbase/button() if the BOOPSI button gadget couldnt be created.
"nbut" will also be raised by buttonbase/render() if NewObjectA() fails to create the gadget.
"nbut" will also be raised by newimagebutton/button() if you don't supply an image to the gadget.

<< Start

1.12 buttonbase.m/Attributes

Common Attributes

These attributes are common to all subclasses of buttonbase:

NB_SELECTED [ISG]

Whether this gadget is selected. This attribute is ignored if the button is neither toggle or push type.
Gadget state can be changed with set() even while it is disabled. (Default: FALSE)

NB_DISABLED [ISG]

Whether this gadget is disabled. Disabled gadgets are properly ghosted in a Style Guide-compliant way. (Default: FALSE)

NB_RESIZEX [I..]

Whether this gadget can resize in width. (Default: FALSE)

NB_RESIZEY [I..]

Whether this gadget can resize in height. (Default: FALSE)

NB_TOGGLE [I..]

Set this to TRUE for a push-on/push-off type button. This tag will take precedence over NB_PUSH. (Default: FALSE)

NB_PUSH [I..]

Set this to TRUE for a push-on-only type button. This type button can only be deselected with set(). This tag is ignored if NB_TOGGLE is TRUE. (Default: FALSE)

NB_FRAMETYPE [I..]

The type of frame for the button, currently either BATT_BUTTONFRAME or BATT_THINFRAME (defined in buttonclass.m). Defaults to the former, the standard Gadtools-like button frame.