PopupMenu

COLLABORATORS			
	TITLE : PopupMenu		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		June 24, 2022	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

PopupMenu

1.1 PopupMenu Library V5.x

Visit the PopupMenu homepage!

http://www.algonet.se/~henisak/pm/pm.html

```
                PM_ItemChecked()

                PM_MakeIDListA()

                PM_MakeItemA()

                PM_MakeMenuA()

                PM_OpenPopupMenuA()

                PM_SetItemAttrsA()
Some examples


                SimpleMenu  - A very simple menu.

                Disable     - Shows disabling and enabling.

                StartMenu   - A little start menu with icons.

                BigMenu     - Unlimited submenus?


                Demo        - From here you can run all the demos above.


                Macros      - Some examples of macro usage.
```

1.2 What is this??

```
Introduction

This is The One and Only Popup Menu Library you use when you want a nice
popup menu in your programs!
```

1.3 Requirements

```
PopupMenu.library requires the following:

* Amiga OS 3.0 or higher.

* commodities.library V36 or higher.
```

1.4 Author

```
Feel free to mail your comments, suggestions, and everything else to:

Henrik Isaksson
Garvarvägen 33
950 40 Töre
SWEDEN

Or...

EMail:
 hki@hem1.passagen.se

The old one should still work:
 amiga_rules@hotmail.com

Try it if you don't get a reply from the first...
```

1.5 DISCLAIMER

```
COPYRIGHT
=========

This software is copyrighted by its developer(s). That means that you
are NOT ALLOWED to modify the program(s) and documentation in any way.
Especially you MUST NOT REMOVE the documentation or this text file.

You are NOT allowed to use this software or any part of it for any other
purpose than that one mentioned in its documentation, this also includes
any fonts,  images or samples.  If the developer(s)  did NOT include the
source code of the program(s) in this package you are NOT allowed to de-
compile any part of it.

* You may use this library for free if your product is free to the user
  (FreeWare, Public Domain or similar).
  The only thing I want is that you mail me and tell me where to find
  the program.

* If your application is ShareWare, you should send me the full release
  or a keyfile.

* If your application is a commercial product you'd better ask me first...
```

```
DISTRIBUTION
============

This package is freely distributable.  That means  you are allowed to re-
distribute this package as long as you follow these points:

* You may NOT ADD any files to the archive!
* You may NOT CHANGE any files in this archive!
* You may NOT REMOVE any of the files in this archive!
* You may include the library file, and the preferences editor in your own
  distributions, if you follow the copyright rules stated above.

This package may be freely distributed via BBSs, InterNet/UseNet,
software libraries such as Fred Fish's and Aminet® CD-ROM, and other
similar electronic channels.

Disk magazines and services that charge extra for file transfers may
NOT distribute it without written permission by the developer(s)!


DISCLAIMER
==========

By using this product, you accept the FULL responsibility for any damage
or loss that might occur through its use or the inability to use it. The
developer(s) of the software  and the author and the translators of this
"Copyright Note" can NOT be held responsible.

Some names used in this text are trademarks or registered trademarks.
The use of these names does not imply that they are free.
```

1.6 Popup Menu Style Guide

```
First read the "Amiga User Interface Style Guide".
Then think about this when designing your menus:

* Font sensitivity is automatically handled by popupmenu.library.
  No need to care for that part. :)

* Use the PMMenu macro for the first menu.
  (Except pulldown menus)

* Submenus should not use a title. (unless they need to)
```

1.7 History & Future

```
The history of popupmenu.library:

5.35  Optimized the pen allocation a bit.

5.3 Moved the submenus a bit to the left. Now it looks more like the
  other menus, and it's easier to open the submenus.
  Fixed a bug in pen allocation/freeing.

  Made a few adjustments to the separator bars.

  Fixed bug in "Submenu Delay" code.

  Fixed an enforcer hit in window mode.

  Got rid of a few unnecessary bytes again. (about 1k smaller)

5.2 Fixed delay bug in non-window mode.

  Improved the remapping of magic images.
  Right look and a few hundred percent faster. :)

  Pulldown menus should now work in non-window mode too.

  Fixed the shadows. They were 1 pixel to wide before.

  Added a new tag, PM_Shadowed, to make the text shadowed.

  Changed the PMMenu macro so the titles will use shadowed text.

  Fixed refresh problem in window-mode.

  Made the separator bars look more like MM2.

  And this time the library is actually 1272 bytes smaller!

5.1 Fixed the deadlock bug in non-window mode.

  Added MagicMenu2 Images, and it's special remapping.
  (relative color values)

  Added support for pulldown menus.
  (works only in window-mode at the moment)

5.0 Opening the menu does no longer deactivate the active window.

  The menus can now use the blitter instead of windows, wich is much
  faster. (this has to be activated in the prefs-editor.)

  The menu font is now fetched from the DrawInfo, and not the menu's
  parent window.

  There is now a space between the bottom of the text and the bottom
  of the select bar.

  Clicking the mousebutton very fast should no longer result in a
  menu that doesn't dissapear. (as it should, when you release the
  button)

  Some of the tags send to OpenPopupMenu() has become obsolete.
  (The preferences has taken over)

  The input method has changed, so the library now requires
  commodities.library (wich is a disk library).

  The menu shadows now look like MagicMenu2 shadows.
  (the size increases for each submenu opened)

  Starting with this version, kickstart 3.0 is required. At least for
  the non-window mode. In window mode it _may_ still work with 2.0,
  but it's not tested. So if you have ks 2.0, I would appreciate if
  you could try it out.

  Fixed a bug. Submenus could appear at the bottom of the screen
  when the mouse was moved quickly over an item.

  Added a new function, PM_AlterState().

4.3 Fixed a bug. (read from adress 0)

4.2 Now the menus can have shadows.

4.1 Images are now remapped correctly.

  Small changes for colour and image prefs.

4.0 More bugs fixed.

  A new demo, MenuVerify, shows how to use popupmenu.library with
  IDCMP_MENUVERIFY.

3.6 Bug fix. (never released, i got more bug reports...)

3.5 Now checks for the file ENV:PopupMenu.cfg, and if it exists,
  loads it, and replaces the default settings.

  Bug in PM_SubMenuTimer fixed.

  PM_Code added. Read this!

3.0 Added PM_GetItemAttrs(), PM_SetItemAttrs(), PM_IsChecked
  and PM_FindItem.

  Added the tags PM_Left and PM_Top.

  Added a few more demos.

  Changed the naming of functions and macros to avoid interfering
  with other libraries.

2.0 Entirely rewritten and is now a shared library!

1.3 Added submenu support

1.2 Lots of new flags, and checkable menuitems

1.1 OpenPopupMenuPos()

1.0 First release
```

```
The future:

1. Finish the prefs editors.

2. Datatype loading of images.

3. Font settings. (maybe)

4. Replacement for BGUI's popbuttonclass using popupmenu.library.

5. Standard icons for common menu items.

6. Add support for multiple selections.

7. Mail me if there's anything else!
```

1.8 "

```
 Special thanks goes to:

 Thanks to Stefan Sommerfeld for his very good bugreports, help with the
 SwapBitsClipRectRastPort() bit, and the remapping code (too bad I coudn't
 use it ;( ).

 Thanks to Trond Werner Hansen for helping me with the shadows!
 (Heja Norge!)

 Thanks to Mario Cattaneo for all the MagicMenu2 images, the other
 MM2 specs and a lot of good advices!

 Thanks to all the rest of you!
```

1.9 PM_MakeMenuA()

```
                  NAME
        PM_MakeMenu -- Create a new menu list.

   SYNOPSIS
        menu = PM_MakeMenuA(taglist);
        d0               a1

        struct PopupMenu *PM_MakeMenuA(struct TagItem *tags);

  menu = PM_MakeMenu(tag1, ...);

  struct PopupMenu *PM_MakeMenu(ULONG, ...);

   FUNCTION
        This function is used to link menu items returned by

                PM_MakeItemA()
                .
   INPUTS
        taglist - pointer to a taglist listing your menu items.

   TAGS
        PM_Item - pointer to a menuitem returned from
                PM_MakeItemA()
                .

   RETURNS
        Returns a pointer to a list of items if successful.

   SEE ALSO

                PM_MakeItemA()
```

1.10 PM_MakeItemA()

```
                   NAME
        PM_MakeItem -- Create a new menu item.

   SYNOPSIS
        menu = PM_MakeItemA(taglist);
        d0                a1

        struct PopupMenu *PM_MakeItemA(struct TagItem *tags);

  menu = PM_MakeItem(tag1, ...);

  struct PopupMenu *PM_MakeItem(ULONG, ...);

   FUNCTION
        This function is used to create a new menu item to be passed to

                PM_MakeMenuA()
                , for linking.

   INPUTS
        taglist - pointer to a taglist listing your menu items.

   TAGS
        PM_Title        (STRPTR) Pointer to the menu text you want.

        PM_UserData     (ULONG) Anything of your choice, can be used to
                        identify the item when it is selected. The value
                        stored here will be returned from
                PM_OpenPopupMenuA()
                                       when the user selects this item.

        PM_ID           (ULONG) An ID number, only needed if you want to
                        be able to read or change the attributes of this
                        item later. (for example, to find out if an item
                        is checked)

        PM_Sub          (struct PopupMenu *) A pointer to a menu list
```

```
                       returned from
                PM_MakeMenuA()
                . The item
                       will automatically get an arrow to the right
                       showing that it has a sub menu.

        PM_Flags       (ULONG) Used internally. Do not use this tag!

        PM_NoSelect    (BOOL) Make the item unselectable.

        PM_FillPen     (BOOL) Draw the item title in FILLPEN.

        PM_Checkit     (BOOL) Leave some space for a checkmark.

        PM_Checked     (BOOL) Put a checkmark to the left of the item.

        PM_Italic      (BOOL) Draw the text in italic.

        PM_Bold        (BOOL) Make the text bold.

        PM_Underlined  (BOOL) Underline the text.

        PM_WideTitleBar (BOOL)
        PM_TitleBar    (BOOL) Draw a horizontal separator instead of the
                       text.

        PM_ShadowPen   (BOOL) Draw the text in SHADOWPEN color.

        PM_ShinePen    (BOOL) Draw the text in SHINEPEN color.

        PM_Exclude     (struct PM_IDLst *) List of items to be selected
                       or unselected when this item gets selected.
                       The list should be created with
                PM_MakeIDListA().
                        PM_Disabled    (BOOL) Makes the item unselectable, and  ↩
                           draws a
                       disable pattern over the item.

        PM_ImageSelected (struct Image *)
        PM_ImageUnselected (struct Image *)
                       Specifies an image to be rendered under the item
                       title.

        PM_IconSelected (struct Image *)
        PM_IconUnselected (struct Image *)
                       Specifies an image to be rendered to the left of the
                       item title.

        PM_AutoStore   (BOOL *) A pointer to a BOOL that will reflect the
                       state of the checkmark. The best way to find out
                       if an item is checked or not.

        PM_TextPen     (ULONG) A pen number for the text. You are
                       responsible for allocating/deallocating a pen
                       yourself.

  PM_Shadowed    (BOOL) Give the the text a shadow using SHADOWPEN.
```

```
   RETURNS
        Returns a pointer to an item if successful.

   SEE ALSO

                PM_MakeMenuA()

                PM_MakeIDListA().

                PM_OpenPopupMenuA()
```

1.11 PM_OpenPopupMenuA()

```
                   NAME
        PM_OpenPopupMenuA -- Open a popup menu.

   SYNOPSIS
        userdata = PM_OpenPopupMenuA(prevwnd, taglist);
        d0                        a1       a2

        ULONG PM_OpenPopupMenuA(struct Window *prevwnd, struct TagItem *tags);

  userdata = PM_OpenPopupMenu(prevwnd, tag1, ...);

  ULONG PM_OpenPopupMenu(struct Window *, ULONG, ...);

   FUNCTION
        This function is used to open a popup menu based on an item list
        created with
                PM_MakeMenuA()
                .

   INPUTS
        prevwnd - pointer to parent window, used to find out screen, font
                  and other drawing attributes.
        taglist - pointer to a taglist of menu options.

   TAGS
        PM_Menu             (struct PopupMenu *) Pointer to a menu list
                            created by
                PM_MakeMenuA()
                .

        PM_RecessSelected   OBSOLETE!
        PM_WideSelectBar    OBSOLETE!
        PM_Compact          OBSOLETE!
        PM_SubMenuTimer     OBSOLETE!
        PM_OldLook          OBSOLETE!
        PM_SameHeight       OBSOLETE!
        PM_CheckMark        OBSOLETE!
        PM_ExcludeMark      OBSOLETE!
        PM_SubMenuMark      OBSOLETE!
        PM_SmartRefresh     OBSOLETE!
        PM_Left              (ULONG) Horizontal position of the menu, relative
                to the menus left edge. (V3)

        PM_Top               (ULONG) Vertical position of the menu, relative
                to the menus top edge. (V3)

  PM_Code        (UWORD) The contents of the Code field of the
          IntuiMessage structure. Used to find out if the
          mousebutton was pressed or released, so the user
          can specify in the preferences if she/he want the
          menu to open when the button is pressed or
          released. Must always be specified!

  PM_PullDown      (BOOL) Turn the menu into a pulldown menu. (V5.1)

   RETURNS
        Returns the value of UserData of the selected item, if no item
        was selected, NULL is returned.

   SEE ALSO

                PM_MakeMenuA()
```

1.12 PM_MakeIDListA()

```
                   NAME
        PM_MakeIDListA -- Create a list of ID's for exclusion/inclusion.

   SYNOPSIS
        list = PM_MakeIDListA(taglist);
        d0                 a1

        struct PM_IDLst *PM_MakeIDListA(struct TagItem *tags);

  list = PM_MakeIDList(tag1, ...);

  struct PM_IDLst *PM_MakeIDList(ULONG, ...);

   FUNCTION
        This function is used to create a list of ID's that is used to
        tell wich items an item should include, exclude, reflect or
        inverse reflect.

   INPUTS
        taglist - pointer to a taglist.

   TAGS
        PM_ExcludeID   (ULONG) ID of a item that should be unselected when
                       when this item is selected.

        PM_IncludeID   (ULONG) ID of a item that should be selected when
                       when this item is selected.

        PM_ReflectID   (ULONG) ID of a item that should copy the state
```

```
                    of this item, when it gets selected/unselected.

       PM_InverseID   (ULONG) ID of a item that should copy the inverse
                      state of this item, when it gets selected/
                      unselected.
                      Useful if you want to make sure only one of two
                      items is selected at a time.

   RETURNS
       Returns a pointer to a list of id's if successful.

   SEE ALSO

              PM_MakeItemA()
```

1.13 PM_FreePopupMenu()

```
                NAME
       PM_FreePopupMenu -- Free a menu list created by
              PM_MakeMenuA()
              .

   SYNOPSIS
       PM_FreePopupMenu(popupmenu);
                        a1

       void PM_FreePopupMenu(struct PopupMenu *);

   FUNCTION
       This function is used to free the list of menu items created by

              PM_MakeItemA()
              , and
              PM_MakeMenuA()
              .

   INPUTS
       popupmenu - pointer to a popup menu to free.

   SEE ALSO

              PM_MakeItemA()

              PM_MakeMenuA()
```

1.14 PM_SetItemAttrsA()

```
                NAME
  PM_SetItemAttrsA -- Specify attribute values for an object. (V3)

   SYNOPSIS
  result = PM_SetItemAttrsA(item, tags);
```

```
D0                     A2    A1

ULONG PM_SetItemAttrsA(struct PopupMenu *, struct TagItem *);

result = PM_SetItemAttrs(item, tag1, ...);

ULONG PM_SetItemAttrs(struct PopupMenu *, ULONG, ...);

 FUNCTION
Specifies a set of attribute/value pairs with meaning as
defined in libraries/pm.h.
item can be directly taken from
              PM_FindItem()
               as the input is
checked against NULL pointers.

 EXAMPLE

struct PopupMenu *menu;

....
/* Initialize the menu... */
....

PM_SetItemAttrsA( PM_FindItem( menu, itemid ),
  PM_Checkit, TRUE,
  PM_Checked, TRUE,
  TAG_DONE);

 INPUTS
item = pointer to a popup menu item.
tags = array of TagItem structures with attribute/value pairs.

 RESULT
Returns the number of successfully changed attributes.

 SEE ALSO
```

1.15 PM_GetItemAttrsA()

```
                 NAME
PM_GetItemAttrsA -- Get attribute values for an object. (V3)

 SYNOPSIS
result = PM_GetItemAttrsA(item, tags);
D0                        A2    A1

ULONG PM_GetItemAttrsA(struct PopupMenu *, struct TagItem *);

result = PM_GetItemAttrs(item, tag1, ...);

ULONG PM_GetItemAttrs(struct PopupMenu *, ULONG, ...);

 FUNCTION
```

```
Used to get attributes from an item.
item can be directly taken from
              PM_FindItem()
               as the input is
checked against NULL pointers.

 EXAMPLE

struct PopupMenu *menu;
struct Image *image;
BOOL checked;

....
/* Initialize the menu */
....

      PM_GetItemAttrsA(PM_FindItem(menu, itemid),
  PM_SelectImage, &image,
  PM_Checked, &checked,
  TAG_DONE);

 INPUTS
item = pointer to a popup menu item.
tags = array of TagItem structures with attribute/value pairs.

 RESULT
Returns the number of successfully copied attributes.

 SEE ALSO
```

1.16 PM_FindItem()

```
 NAME
PM_FindItem -- Find an item in a popupmenu list. (V3)

 SYNOPSIS
item = PM_FindItem(menu, id);
D0                 A1    D1

struct PopupMenu *PM_FindItem(struct PopupMenu *, ULONG);

 FUNCTION
Find the pointer to an item using the ID number.

 INPUTS
menu = pointer to a popup menu list.
id   = ID number (PM_ID).

 RESULT
Returns a pointer to the found item, or NULL if unsuccessful.

 SEE ALSO
```

1.17 PM_ItemChecked()

```
   NAME
  PM_ItemChecked -- Find out if an item is checked. (V3)

   SYNOPSIS
  item = PM_ItemChecked(menu, id);
  D0                    A1    D1

  BOOL PM_ItemChecked(struct PopupMenu *, ULONG);

   FUNCTION
  Fast way to find out if an item is checked using the item ID.

   INPUTS
  menu = pointer to a popup menu list.
  id   = ID number (PM_ID).

   RESULT
  TRUE (-1L) if the item is checked, FALSE (0L) if not checked,
  PMERR (-5L) if the ID was not found in the list.

   SEE ALSO
```

1.18 PM_AlterState()

```
   NAME
  PM_AlterState -- Change a lot of items at the same time. (V5)

   To be continued... (next release)
```

1.19 SimpleMenu

```
This menu shows how a context sensitive Trashcan popup menu, could look
like, in a workbench replacement application.
The menu opens on either a click with the left or the right mouse button.

Run and click the mouse in the window!
See the source!
```

1.20 StartMenu

```
This example shows how to put icons in your menus.
(Theese icons are NOT remapped, and will look best with a MagicWB palette)
The menu opens when the mousepointer touches the bottom of the screen.

Run and move the mouse to the bottom of the screen!
See the source!
```

1.21 Disable

```
This examples shows how disable/enable menu items, and how to see if an
item is checked or not.

Run and play with the menu!
See the source!
```

1.22 BigMenu

```
This example is kind of a hack.
It doesn’t free all the memory it uses, and crashes eventuallty.
It opens a menu with a submenu with a submenu with another submenu, and so
on... It’s not really possible to an unlimitet number of submenus, and it
has been reported to act strange at the 53rd submenu... :)
(If you want to see it crash earlier, try to reduce the stack size. :) )

Run, but don’t open too many submenus!!
See the source!
```

1.23 Demo

```
This example can be used to start all the other demos, and it also shows one
way to have two different menus. The menu opens only when you hold down the
_right_ mousebutton. The first menu is in the dragbar, and the second is in
anywhere below the dragbar.

Run and try it out!
```

1.24 Macros Example

```
 struct PopupMenu *menu;

 /*
  Creating a menu with the macros...
 */

 menu = PMMenu("The title of the menu"),
  End;

 /*
  Creating a menu like the one above, but with items...
 */

 menu = PMMenu("The menu title"),
    PMItem("The item text"),
      /* Tags for the item */
    End,
    PMItem("The item text (2)"),
      /* Tags for the item */
    End,
  End;

 /*
  Creating a sub-menu for an item...
 */

 menu = PMMenu("The menu title"),
    PMItem("Item with submenu"),
      PMSubMenu("This is the submenu"),
        /* Here are the items... */
      End,
    End,
  End;

 /*
  Creating an exclude ID list for PMExclude
 */

  PM_Exclude,
    PMExl ExID(1),
      ExID(2),
  End,
```