Help topics

MainMenu

Pavel, 31 January 2012 (created 14 November 2011)

no tags

PUM - popup menu
WinClip

PUM - popup menu

Pavel, 31 January 2012 (created 14 November 2011)

no tags

PUM is an AutoHotkey_L module developed for creating and using customizable owner-draw popup menus

Download link (.zip 13.3 KB)

Has three classes to create and manipulate with the menus:
PUM - main manager class, creates menus
PUM_Menu - class representing popup menu
PUM_Item - class representing popup menu's item

Usage Example:

Code:

```
#NoEnv
#Include PUM_API.ahk ;include this first
#Include PUM.ahk

; parameters of the PUM object, the manager of the menus
pumParams := { "SelMethod" : "fill"            ;item selection method, may be frame,fill
                ;~ ,"selTColor" : -1         ;selection text color
                ;~ ,"selBGColor" : -1              ;selection background color, -1 means invert current color
                ,"oninit"      : "PUM_out"      ;function which will be called when any menu going to be opened
                ,"onuninit"    : "PUM_out"     ;function which will be called when any menu going to be closing
                ,"onselect"    : "PUM_out"     ;;function which will be called when any item selected with mouse (hove
                ,"onrbutton"   : "PUM_out"   ;function which will be called when any item right clicked
                ,"onmbutton"   : "PUM_out"  ;function which will be called when any item clicked with middle mouse but
                ,"onrun"       : "PUM_out"      ;function which will be called when any item clicked with left mouse b
                , "onshow" : "PUM_out"      ;function which will be called before any menu shown using Show method
                , "onclose" : "Pum_out"        ;function called just before quitting from Show method
                ,mnemonicCMD : "select"}

;PUM_Menu parameters
menuParams1 := { "bgcolor" : 0x36311f   ;background color of the menu
            , "iconssize" : 32          ;size of icons in the menu
            , "tcolor" : 0xafc9d3 }     ;text color of the menu items
menuParams2 := { "bgcolor" : 0x1C3150
            , "iconssize" : 16
            , "tcolor" : 0xFFFFFF }

;create an instance of PUM object, it is best to have only one of such in the program
pm := new PUM( pumParams )

;creating popup menu, represented by PUM_Menu object with given parameters
```

```
menu := pm.CreateMenu( menuParams1 )

;create a three othe menus
newmenu1 := pm.CreateMenu( menuParams2 )
newmenu2 := pm.CreateMenu( menuParams2 )
newmenu3 := pm.CreateMenu( menuParams2 )

;adding submenu items to the first menu, item SubItem1 will open "newmenu1" menu, and SubItem2 will open "newmenu2"
menu.add( { "name" : "SubItem1", "submenu" : newmenu1, "icon" : "shell32.dll:8"  } )
menu.add( { "name" : "SubItem2", "submenu" : newmenu2, "icon" : "shell32.dll:8" } )
menu.Add()  ;adding separator
;adding a submenu item "SubItem3" to the "newmenu2" menu, which opens "newmenu3" menu
newmenu2.add( { "name" : "SubItem3", "icon" : "shell32.dll:8", "submenu" : newmenu3 } )

;adding five items to the first menu
loop, 5
  menu.Add( { "name" : "i&tem" A_Index
            , "bold" : 1
            , "icon" : "shell32.dll:" A_index+20 } )
;adding five items to the newmenu1
loop,5
        newmenu1.add( { "name" : "item" A_index, "icon" : "shell32.dll:3" A_index } )
;adding five items to the newmenu2
loop,5
  newmenu2.add( { "name" : "item" A_index, "icon" : "shell32.dll:4" A_index } )
;adding five items to the newmenu3
loop,5
  newmenu3.add( { "name" : "item" A_index, "icon" : "shell32.dll:2" A_index, disabled : 1 } )

;showing the first menu at the center of screen (~)
if ( item := menu.Show( A_ScreenWidth/3, A_ScreenHeight/3 ) )
        msgbox % "Choosen item: " item.name

;Destroying all menus/items created
;Use Destroy() method for the PUM_Menu object if you want to destroy specific menu
pm.Destroy()
ExitApp
return

PUM_out( msg, obj )
{
  if ( msg = "onselect" )
  {
    rect := obj.GetRECT()
    CoordMode, ToolTip, Screen
    tooltip,% "Selected: " obj.name,% rect.right,% rect.top
  }
  if ( msg ~= "oninit|onuninit|onshow|onclose" )
    tooltip % "menu " msg ": " obj.handle
  if ( msg = "onrbutton" )
    tooltip % "Right clicked: " obj.name
  if ( msg = "onmbutton" )
    tooltip % "Middle clicked: " obj.name
  if ( msg = "onrun" )
    tooltip % "Item runned: " obj.name
}
```

PUM_Item

Pavel, 31 January 2012 (created 14 November 2011)

PUM_Item is class representing popup menu's item created through **PUM_Menu**.Add() method

An item may have associated menu as submenu

no tags

Example:

```
ItemParams := { "name" : "SomeItemName"
            , "bold" : 1
            , "icon" : "shell32.dll:20" }
oPUM := new PUM()
menu := oPUM.CreateMenu( MenuParams )
item := menu.Add( ItemParams  )
menu.Show( 100, 100, "hcenter animlr" )
item.Destroy()
```

PUM_Item parameters. Change them through SetParams() method only

Parameter Name	Default Value	Possible Values	Description
issep	0	0,1	1 if item is separator, 0 otherwise
name	<item_title>	str	item's name
bold	0	0,1	if 1, item will be shown in bold in the menu
icon	0	str	a path to or handle of the icon, may be in form path:icon_index, any pic files
break	0	0,1,2	if 1 - will break menu after this item, and next one will start new column if 2 - menu break will be shown in form if vertical line
submenu	0	**PUM_Menu** object or popup menu handle	if present, item will open submenu on hover
tcolor	""	RGB	text color of item, if empty, color of the parent menu will be used
bgcolor	""	RGB	background color of item, if empty, color of the parent menu will be used
noPrefix	0	0,1	if 1, will not interpret "&" mnemonics
uid	""	str	custom ID of the item, may be any string/integer
disabled	0	0,1	if 1, item will look disabled & and not clickable
iconUseHandle	0	0,1	if this flag is True, and icon handle passed through SetParams in "icon" field, then this handle will not be destroyed when item deleted

Read-only fields, do not change them:

Name	Description
alive	not 1 if item destroyed
id	inner id of the item, always unique
menu	**PUM_Menu** object, this item belongs to
assocMenu	**PUM_Menu** object, associated with that item as submenu
hFont	handle to the font for this item
hIcon	handle to the associated with item icon
hotCharCode	ascii code of the character to check when user press any key while menu opened

PUM_Item methods:

Name	In	Out	Description

Name	In	Out	Description
SetParams()	objItemParams	-	updates required parameters of the item, only parameters presented in objItemParams will be updated
RemoveSubMenu()	-	-	detach associated with item menu, will not destroy item or menu
DestroySubMenu()	-	-	destroy associated with item menu
GetPos()	-	0-based Position	returns 0-based position of this item in menu
GetRect()	-	object	returns RECT object representing bounding rectangle of the current menu item expressed in screen coordinates object has the following fields: left,right,top,bottom
Update()	-	-	Updates item's parameters. Useful if you changed anything directly, without using *SetParams*() method
Destroy()	-	-	destroy item and associated menu if any
Detach()	-	-	destroy item and removes menu without destroying it
GetTColor()	-	RGB	Returns RGB color value used by this item as text color
GetBGColor()	-	RGB	Returns RGB color value used by this item as background color
GetIconHandle()	-	hIcon	Returns handle to the icon used by this item to draw icon

PUM_Menu

Pavel, 31 January 2012 (created 14 November 2011)

no tags

PUM_Menu is class representing popup menu created through **PUM**.CreateMenu() method
A menu can be associated with item, in which case it will have "owner" field representing **PUM_Item** object, it belongs to
By destroying **PUM_Menu** all items it contain will be also destroyed

Example:

```
MenuParams := { "bgcolor" : 0x36311f
            , "iconssize" : 16
            , "tcolor" : 0xafc9d3 }
oPUM := new PUM()
menu := oPUM.CreateMenu( MenuParams )
menu.Add( "item" )
menu.Show( 100, 100, "hcenter animlr" )
menu.Destroy()
```

PUM_Menu parameters. Change them through SetParams() method only

Parameter Name	Default Value	Possible Values	Description
tcolor	windows default	RGB	text color of the items in the menu
bgcolor	windows default	RGB	RGB background color of the menu
nocolors	0	0,1	if 1, will be used default color for menu's background and item's text color
noicons	0	0,1	if 1, icons will not be shown in the menu
notext	0	0,1	if 1, text will not be shown for the item, should not be used with "noicons"
iconssize	32	int preferably 16,32	icon size for items
textoffset	5	int	gap between icon and item's text in pixels
maxheight	0	int	max height of the menu, scroll will be added if menu is bigger
xmargin	3	int	margin for the left and right of item boundary
ymargin	3	int	margin for the top and bottom of item boundary

Parameter Name	Default Value	Possible Values	Description
textMargin	5	int	pixels amount which will be added after the text to make menu look pretty

Read-only fields, do not change them:

Name	Description
alive	not 1 if menu destroyed
handle	handle of the popup menu
objPUM	reference to the parent PUM object
owner	reference to the **PUM_Item** object, with which this menu associated if any

PUM_Menu methods:

Name	In	Out	Description
SetParams()	objMenuParams	-	updates required parameters of the menu, only parameters presented in objMenuParams will be updated

Add()	objItemParams
	prevItem(opt)
	fByPos(opt)
		PUM_Item on success
		0 otherwise
			adds new item to menu
			objItemParams - parameters of the new item
			prevItem - position or ID of the item, after which new one will be added
			fByPos - if 1, prevItem is 0-based position, if 0 - prevItem is **PUM_Item**.id after which new item will be added
			to add item to the end of the menu, pass -1 instead of prevItem (by default)
			if objItemParams empty, separator will be added
			if objItemParams - any string, item with that name will be added

GetItems()	-	**PUM_Item** array	returns an array of items menu currently has
EndMenu()	-	-	immediately close any popupmenu menu on screen
IsMenu()	-	1 or 0	returns 1 if current menu is still exist (not destroyed)
GetItemByPos()	0-based position	**PUM_Item**	returns **PUM_Item** object representing item at specific position in the menu
Count()	-	number of items in menu	returns current number of items in the menu
GetTColor()	-	RGB	returns current text color used by this menu
GetBGColor()	-	RGB	returns current background color used by this menu
Destroy()	-	-	destroys the menu and all its items, and all menus associated with this items
Detach()	-	-	detach this menu from the item, it belongs to. Will not destroy anything
Show()	x y flags	**PUM_Item** if any selected 0 otherwise	show this menu at X,Y on screen if any item was selected, this function returns it's **PUM_Item** object Flags is space delimeted list of options listed below

Flags, used in PUM_Menu.Show():

Name	Description
context	will show context popup menu above one currently opened, menu will not be shown if you use this flag when no other menu is shown
hcenter	Centers the shortcut menu horizontally relative to the coordinate specified by the x parameter.
hleft	(Default) Positions the shortcut menu so that its left side is aligned with the coordinate specified by the x parameter.

Name	Description
hright	Positions the shortcut menu so that its right side is aligned with the coordinate specified by the x parameter.
vbottom	Positions the shortcut menu so that its bottom side is aligned with the coordinate specified by the y parameter.
vtop	(Default) Positions the shortcut menu so that its top side is aligned with the coordinate specified by the y parameter.
vcenter	Centers the shortcut menu vertically relative to the coordinate specified by the y parameter.
animlr	Animates the menu from left to right.
animrl	Animates the menu from right to left.
animtb	Animates the menu from top to bottom.
animbt	Animates the menu from bottom to top.
noanim	Displays menu without animation.

PUM

Pavel, 31 January 2012 (created 14 November 2011)

no tags

PUM is manager class, through which you can create **PUM_Menu**'s
It is best to have only one **PUM** instance in your program
By destroying **PUM** object all menus/items created through it also destroyed.

Example:

```
PUMParams := { "SelMethod" : "fill"
                ,"selTColor" : -1
                ,"selBGColor" : -1}
oPUMMenu := new PUM( PUMParams )
menu := oPUMMenu.CreateMenu()
menu.Add( "item" )
menu.Show( 100, 100 )
oPUMMenu.Destroy()
```

PUM parameters. Change them through SetParams() method only

Parameter Name	Default Value	Possible Values	Description
selMethod	fill	fill frame	a form of selection which appears when user hover mouse cursor over menu item
selBGColor	windows default	any RGB	background color of selected item, -1 means current color will be inverted
selTColor	windows default	any RGB	text color of selected item, -1 means current color will be inverted
frameWidth	1	any integer	width of select frame when selMethod = "frame"
mnemonicCmd	run	select run	indicate what happens when user press any key associated with menu item as hotkey "run" means the item will be runned as if user clicked it "select" means it will be selected/highlighted
oninit	""	any valid func name	name of the function, which will be runned whenever any menu is opening the first parameter of the target function will be equal to "oninit" the second parameter will be the **PUM_Menu** instance of the menu, which is openin
onuninit	""	any valid func name	name of the function, which will be runned whenever any menu is closing the first parameter of the target function will be equal to "onuninit" the second parameter will be the **PUM_Menu** instance of the menu, which is closing
onselect	""	any valid func name	name of the function, which will be runned whenever any item in menu is selected/hovered the first parameter of the target function will be equal to "onselect" the second parameter will be the **PUM_Item** instance of the selected item

Parameter Name	Default Value	Possible Values	Description
onrbutton	""	any valid func name	name of the function, which will be runned whenever user pressed right mouse button on item the first parameter of the target function will be equal to "onrbutton" the second parameter will be the **PUM_Item** instance of the targeted item
onmbutton	""	any valid func name	name of the function, which will be runned whenever user pressed middle mouse button on item the first parameter of the target function will be equal to "onmbutton" the second parameter will be the **PUM_Item** instance of the targeted item
onrun	""	any valid func name	name of the function, which will be runned whenever user run any item the first parameter of the target function will be equal to "onrun" the second parameter will be the **PUM_Item** instance of the targeted item
onshow	""	any valid func name	name of the function, which will be runned each time **PUM_Menu**.Show() method called but before menu is appeared the first parameter of the target function will be "onshow" the second parameter will be the **PUM_Menu** instance of the menu which going to be shown If target function exist and it returned 0, menu will not be shown
onclose	""	any valid func name	name of the function, which will be runned each time **PUM_Menu**.Show() method called but after the menu is closing the first parameter of the target function will be "onclose" the second parameter will be the **PUM_Menu** instance of the menu which was shown

PUM Methods

Method Name	In parameters	Returns	Description
SetParams()	objPUMParams	-	update required parameters of the PUM object, only parameters presented in objParams will be updated
GetMenu()	menuHandle	**PUM_Menu** on success 0 otherwise	retrieve **PUM_Menu** object through it's handle
GetItemByUID()	uid	**PUM_Item** on success 0 otherwise	retrieve **PUM_Item** object through it's uid (user's custom ID)
GetItemByID()	id	**PUM_Item** on success 0 otherwise	retrieve **PUM_Item** object through it's id (can be retrieved through **PUM_Item**.id after the item was added to menu)
CreateMenu()	objMenuParams	**PUM_Menu** on success 0 otherwise	create new **PUM_Menu** object using given parameters
Destroy()	-	-	destroys PUM object and all menus/items created through it