Buddy Menu 1.2

Buddy Menu is a script xtra that makes system context/pop-up menus for Windows and OSX. It can also be used in Director Shockwave. Director and Authorware are both supported.
The xtra has no visual interface – all menus are created and displayed by scripting.

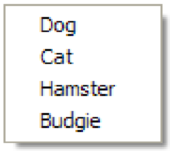
The simplest way to create and display a menu is the bmMenu command.

```
bmMenu list Items, integer Flags
```

The function takes a list of the items to be displayed in the menu, and a Flags argument to modify the behaviour of the menu. It returns the text that the user selected. For example;

```
pet = bmMenu( [ "Dog", "Cat", "Hamster", "Budgie"], 0 )
```

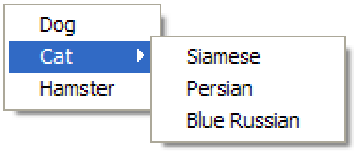
shows the following menu:



and returns the item the user selected. If the user does not select an item, an empty string is returned.

You can attach a sub menu by embedding a list after the item.

```
pet = bmMenu( [ "Dog", "Cat", [ "Siamese", "Persian", "Blue Russian"], "Hamster"], 0 )
```



Sub menus can also be added to items on sub menus.

To include a separator, pass in an empty string.

The menu will be displayed at the current mouse position. You can specify a position to display the menu at by using the bmMenuAt function, which adds X and Y co-ordinates.

```
bmMenuAt list Items, integer Flags, integer X, integer Y
```

There are four Flags to specify the alignment of the menu to the mouse position, or the specified X and Y arguments.

1	align right
2	align center horizontally
4	align bottom
8	align center vertically

You can add the Flags together if you want to use more than one. For example, to align the menu to the bottom right, use 1+4 as the Flag.

There is one flag to specify the return style.
16 return as list
The return will be formatted as a list, and include all the menus selected. For example:

```
bmMenu( [ "Arial", [“Bold”, “Italic”], "Times", [ "Bold", "Italic" ] ], 16 )
```

will return [“Arial”, “Bold”] if the user selects the Arial Bold menu. You will need to use this flag if you have more than one menu item with the same name. If you don’t use this flag, then in the above case, the return will be “Bold”, but you won’t be able to determine which font was selected.

This function is self-contained – the menu is created, shown and destroyed in one call.

There are other functions which give you more control over the menus – to disable and check items, and to modify existing menus. The basic procedure is to use bmCreateMenu to create an empty menu, use bmAppendItem or bmInsertItem to add menu items, bmShowMenu to display the menu, then finally bmDestroyMenu to destroy the menu when you are finished with it. Functionally,

```
bmMenu( [ "one", "two", "three" ], 0 )
```

is identical to

```
menu = bmCreateMenu()
bmAppendItems( menu, [ "one", "two", "three" ] )
bmShowMenu( menu, 0 )
bmDestroyMenu( menu )
```

bmCreateMenu creates a new empty menu. It returns a menu handle, which is then used in subsequent Buddy Menu functions. You can create as many different menus as you need to. The menu will remain valid until you call bmDestroyMenu.

```
menu = bmCreateMenu()
```

Once you have created a menu, you need to add items to it. There are six functions to do this:

```
bmAppendItem integer Menu, string Item
bmAppendItems integer Menu, list Items
bmInsertItem integer Menu, string Item, string beforeItem
bmInsertItems integer Menu, list Items, string beforeItem
bmInsertItemAt integer Menu, string Item, integer Pos
bmInsertItemsAt integer Menu, list Items, integer Pos
```

bmAppendItem adds a new item to the end of the menu. bmAppendItems add a list of items to the end of the item. The list of items is in the same format as bmMenu – you can add submenus by using embedded lists.
bmInsertItem inserts a new menu item. You pass in the text of the item you want to insert the item before. bmInsertItems inserts a list of items.
bmInsertItemAt also inserts a new menu item. You pass in the position where you want the new item to be, eg 1 to be at the start of the menu, 3 to be the third item. bmInsertItems inserts a list of items.

Once you have a menu with items, you can use other functions to modify the items.

```
bmEnableItem integer Menu, string Item
bmDisableItem integer Menu, string Item
bmItemDisabled integer Menu, string Item
bmCheckItem integer Menu, string Item
bmUncheckItem integer Menu, string Item
bmItemChecked integer Menu, string Item
```

bmDisableItem disables the menu item so that it can not be selected. Use bmEnableItem to enable it again. You can use bmItemDisabled to check whether or not an item is disabled.

bmCheckItem checks the menu item so that it has a check mark beside it. bmUncheckItem unchecks it again. You can use bmItemChecked to check whether or not an item is checked. Note that the system does not handle the checking and unchecking of the item – you need to keep track of whether an item is checked or unchecked and then change it yourself if the item is selected. For example, if you have a font menu with a Bold item, then the user selecting the Bold item when it is checked means they want to make the text normal, and you will need to do that and uncheck the Bold item.

```
theItem = bmShowMenu( menu, 0 )
if theItem = "Bold" then
  if bmItemChecked( menu, "Bold" ) then -- item is presently checked, so make it unchecked
     bmUnCheckItem( menu, "Bold" )
     makeTextNormal() -- make your text normal
  else -- item is presently unchecked, so make it checked
    bmCheckItem( menu, "Bold" )
    makeTextBold()
  end if
end if
```

Use bmDeleteItem to delete an item.

Use bmChangeItemText to change the text of an existing item.

One menu can be added to another menu as a sub menu using bmAttachSubMenu.

```
menu = bmCreateMenu()
bmAppendItems( menu, ["Cat", "Dog", "Hamster" ] )
dogMenu = bmCreateMenu()
bmAppendItems( dogMenu, [ "Alsation", "Corgi", "Poodle" ] )
bmAttachSubMenu( menu, "Dog", dogMenu )
pet = bmShowMenu( menu, 0 )
```

Use bmDetachSubMenu to remove a sub menu.

When you are finished with your menu, and don't need to use it again, use bmDestroyMenu to destroy the menu and clear the memory it used. Any submenus attached to the menu will also be destroyed. Once you have destroyed a menu, do not use the menu handle again.

Director Shockwave

The xtra is marked as Safe for Shockwave, and is available as a downloadable package. Place the BudMenu.w32, BudMenu.carb and BudMenu.xpku files on to your web server. If you are using MX, place the BudMenu.ppc file there as well. Note that .ppc packages contain Classic xtras, but Buddy Menu is not available as a Classic xtra. Director MX requires that a Classic package be available, so a dummy one is provided.

Then open the xtrainfo.txt file – in MX it is in the 'Director MX' folder, on MX 2004 and D11 in the 'Configuration' folder. You need to add a line to the end of the file:

```
[#namePPC: "Buddy Menu Xtra", #nameW32: "Buddy Menu Xtra.x32", #package:
"http://www.mods.com.au/budmenu/BudMenu"]
```

You need to replace the package location with the location of the package files on your web server.

To make the xtra automatically download, add it to the list of xtras to be embedded. Select the Modify->Movie->Xtras menu and then add Buddy Menu to the list of xtras. Make sure that Buddy Menu is selected in the list and then check the 'Download If Needed' option. You will need to be connected to the internet and have the packages available in the correct location.