Package 'svWidgets'

October 14, 2022

Type Package

Version 0.9-45

Date 2018-06-28

Title Management of GUI Widgets, Windows, and Other GUI Resources

Author Philippe Grosjean [aut, cre]

Maintainer Philippe Grosjean <phgrosjean@sciviews.org>

Depends R (>= 2.7.0)

Imports tcltk, utils, svMisc (>= 0.9-68)

Description High level management of widgets, windows and other graphical resources.

License GPL-2

URL http://www.sciviews.org/SciViews-R

BugReports https://r-forge.r-project.org/tracker/?group_id=194

NeedsCompilation no

Repository CRAN

Date/Publication 2018-06-28 09:47:56 UTC

R topics documented:

svWidgets-package *SciViews GUI API - Widgets & Windows*

Description

High level management of widgets, windows and other graphical resources.

Details

Package:	svWidgets
Type:	Package
Version:	0.9-45
Date:	2018-06-28
License:	GPL 2 or above, at your convenience

Author(s)

Philippe Grosjean

Maintainer: Ph. Grosjean <phgrosjean@sciviews.org>

Img *Manipulate image resources for the GUIs*

Description

Mechanism provided here is very simple and allows for automatic loading of image resources from any package subdirectory. Currently, only Tk images loaded from GIF files are supported... but more formats could be added in the future.

Usage

```
imgAdd(file, type = "gif", img.type = "tkImage", update = FALSE, ...)
imgDel(image)
imgGet(image)
imgNames()
imgType(image, warn = TRUE)

imgRead(dir, type = "gif", img.type = "tkImage")
imgReadPackage(package, subdir = "gui", type = "gif", img.type = "tkImage")

## S3 method for class 'guiImg'
print(x, ...)
```

Arguments

`file`	image file from where to download the resource.
`type`	type of image. Currently, only 'gif' is supported.
`img.type`	the type of image resource to create. Currently, only 'tkImage' is supported and it is a Tcl/Tk resource.
`update`	do we update an already loaded image resource, or not?
`image`	Name of an image.
`warn`	do we issue a warning if the type of image is not recognized?
`dir`	the directory that contains one or more image files to read.
`package`	name of a package from where to load image resources.
`subdir`	subdirectory in the package where the graphical resources are stored. By default, it is the \"gui\" subdirectory.
`x`	an object of class 'guiImg'.
`...`	further arguments (currently not used).

Details

These functions care about (un)loading image resources. A list of these resources is maintained in '.guiImgs' located in the `SciViews:TempEnv` environment.

Value

`imgAdd()` and `imgGet()` return the handle to the newly created image (invisibly for the `imgAdd()`). `imgDel()` returns invisibly `TRUE` if the resource is found and deleted, `FALSE` otherwise. `imgNames()` return the list of all images registered in .guiImgs in the `SciViews:TempEnv` environment. `imgRead()` and `imgReadPackage()` return invisibly the list of image files that are imported as resources.

Author(s)

Philippe Grosjean

See Also

`toolAdd`

Examples

```
## Not run:
## These cannot be run by examples() but should be OK when pasted
## into an interactive R session with the tcltk package loaded

imgNames()
myImg <- imgAdd(system.file("gui", "logoSciViews.gif", package = "svWidgets"))
myImg  # Note that $Tk. is prepended to the name!
imgNames()
imgType(myImg)
## Place that logo in a Tk window
```

```
timg <- winAdd("timg", title = "A Tk window with image", pos ="-40+20")
labImg <- tklabel(timg, image = imgGet(myImg), bg = "white")
tkpack(labImg)
## When the image resource is deleted, it is not displayed any more (no error)
imgDel(myImg)
imgNames()
winDel("timg")
## To read all image resources at once (place this in .Lib.first())
imgReadPackage("svWidgets")
imgNames()
rm(myImg)

## End(Not run)
```

Menu *Conveniently manipulate menus, whatever the window*

Description

These functions provide an unifying way of dealing with menus in R. It is inspired from the winMenuXXX() functions under Windows that allow easy manipulation of custom menus in Rgui.exe. Currently, they support all the functionnalities of winMenuXXX() functions and they bring a convenient similar layer on top of Tk menus to manipulate them in a similar way.

Usage

```
menuAdd(menu, ...)
menuAddItem(menu, item, action, image = "", accel = "", options = "")
menuDel(menu)
menuDelItem(menu, item)
menuNames()
menuItems(menu)
menuType(menu, warn = TRUE)
menuInvoke(menu, item)
menuChangeItem(menu, item, action = "", options = "")
menuStateItem(menu, item, active = TRUE)
menuRead(file = "Menus.txt")
menuReadPackage(package, subdir = "gui", file = "Menus.txt")

## S3 method for class 'guiMenu'
print(x, ...)
```

Arguments

`menu`	name of a menu.
`item`	name of a menu item.
`action`	action the menu triggers (R code).

`image`	name of an image to display at left of the menu item.
`accel`	accelerator (keystroke) to use to trigger this menu item.
`options`	additional options, for instance "disable" to disable the menu at creation.
`warn`	do we issue a warning if the type of menu is not recognized?
`active`	do we enable or disable the menu item?
`file`	a file containing menu specifications to read.
`package`	name of a package from where to load menu specifications.
`subdir`	subdirectory in the package where the menu specifications are stored. By default, it is the "gui" subdirectory.
`x`	an object of class 'guiMenu'.
`...`	further arguments passed to the function.

Details

These functions care about creating, deleting and managing custom menus. Informations and handles to the various menus created with these functions are stored in the .guiMenus variable, located in the `SciViews:TempEnv` environment.

Value

`menuAdd()` and `menuAddItem()` return the handle to the newly created menu/menu item invisibly. `menuDel()` and `menuDelItem()` return invisibly `TRUE` if the resource is found and deleted, `FALSE` otherwise. `menuNames()` returns the list of all menus registered in .guiMenus in the `SciViews:TempEnv` environment. `menuInvoke()` returns invisibly `TRUE` if the menu item was invoked, `FALSE` otherwise. `menuRead()` and `menuReadPackage()` return invisibly the list of menus that are imported and created.

Author(s)

Philippe Grosjean

See Also

`tkMenuAdd`, `imgReadPackage`

Examples

```
## Not run:
## These cannot be run by examples() but should be OK when pasted
## into an interactive R session with the tcltk package loaded
## Run these commands one at a time and look at menus...


### Under RGui and Windows only! ###
## Create menus in Rgui, using a specification file
menuReadPackage("svWidgets")
menuNames()
(menuItems("$ConsoleMain/Testit"))
```

```
## Create menus manually in Rgui
menuAdd("$ConsoleMain/Testit2")
menuAddItem("$ConsoleMain/Testit2", "Trial", "ls()")
menuNames()
(menuItems("$ConsoleMain/Testit2"))
menuStateItem("$ConsoleMain/Testit2", "Trial", FALSE)
menuStateItem("$ConsoleMain/Testit2", "Trial", TRUE)
## Buggy? -> menuChangeItem("$ConsoleMain/Testit2", "Trial", "search()")
(menuItems("$ConsoleMain/Testit2"))


### Under any system supporting Tcl/Tk ###
## Create and manipulate Tk menus
winAdd("tt", title = "A Tk window with menus", pos ="-40+20")
menuAdd("$Tk.tt/Misc")
menuNames()
(menuItems("$Tk.tt/Misc"))  # Still nothing in it
menuAddItem("$Tk.tt/Misc", "List &variables", "print(ls(envir = .GlobalEnv))")
menuAddItem("$Tk.tt/Misc", "Say &yo!", "cat('yo!\n')")

menuDelItem("$Tk.tt/Misc", "Say &yo!")
menuAddItem("$Tk.tt/Misc", "-")
menuAddItem("$Tk.tt/Misc", "&Say yo! twice", "cat('yo! yo!\n')")
(menuItems("$Tk.tt/Misc"))

imgReadPackage("svWidgets")  # Make sure images are loaded
menuAdd("$Tk.tt/Misc/Sub&Menu")
menuAddItem("$Tk.tt/Misc/Sub&Menu", "T&rial", "cat('Trial trigerred!\n')")
menuAddItem("$Tk.tt/Misc", "Tria&l2", "cat('Trial with image and accel!\n')",
image = "$Tk.butOpen", accel = "Ctrl+T")
menuNames()
(menuItems("$Tk.tt/Misc"))
menuStateItem("$Tk.tt/Misc", "Tria&l2", FALSE)
menuStateItem("$Tk.tt/Misc", "Tria&l2", TRUE)
menuStateItem("$Tk.tt/Misc", "Sub&Menu", FALSE)
menuStateItem("$Tk.tt/Misc", "Sub&Menu", TRUE)
menuChangeItem("$Tk.tt/Misc", "Tria&l2", options = "underline = 1")
## This is the way to change binding
tkbind(WinGet("tt"), "<Control-r>", function() menuInvoke("$Tk.tt/Misc", "Tria&l2"))
menuChangeItem("$Tk.tt/Misc", "Tria&l2", action = 'cat("new action for Tria&l2!\n")')
menuInvoke("$Tk.tt/Misc", "Tria&l2")
(menuItems("$Tk.tt/Misc"))
menuDelItem("$Tk.tt/Misc", "Tria&l2")
menuDel("$Tk.tt/Misc")
menuNames()
## The following command generates an error!
(menuItems("$Tk.tt/Misc"))
winDel("tt")

## End(Not run)
```

tkImg *Manipulate image resources for Tcl/Tk GUIs*

Description

These functions read image files on disk and create Tk image resources.

Usage

```
tkImgAdd(file, type = "gif", update = FALSE)
tkImgDel(image)
tkImgRead(dir, type = "gif")
```

Arguments

`file`	image file from where to download the resource.
`type`	type of image. Currently, only 'gif' is supported.
`update`	do we update an already loaded image resource, or not?
`image`	name of an image.
`dir`	the directory that contains one or more image files to read.

Details

These functions should not be called directly, but by means of `imgAdd()`, `imgDel()` or `imgRead()`.

Value

`tkImgAdd()` returns the handle to the newly created image invisibly. `tkImgDel()` returns invisibly `TRUE` if the resource is found and deleted, `FALSE` otherwise. `tkImgRead()` returns invisibly the list of image files that are imported as resources.

Author(s)

Philippe Grosjean

See Also

`imgAdd`, `imgDel`, `imgRead`

tkMenu *Conveniently manipulate Tk menus*

Description

These functions provide an easy way to create and manipulate Tk menus under R. Note that the corresponding `menuXXX()` function also manipulate Tk menus the same way, but are capable of manipulating other menus as well. One should, thus, preferably use `menuXXX()`!

Usage

```
tkMenuAdd(menu, tearoff = FALSE)
tkMenuAddItem(menu, item, action, image = "", accel = "", options = "")
tkMenuDel(menu)
tkMenuDelItem(menu, item)
tkMenuItems(menu)
tkMenuChangeItem(menu, item, action = "", options = "")
tkMenuStateItem(menu, item, active = TRUE)
tkMenuInvoke(menu, item)
tkMenuItemCall(expr)
```

Arguments

`menu`	name of a menu.
`tearoff`	should the menu be detachable?
`item`	name of a menu item.
`action`	action the menu triggers (R code).
`image`	name of an image to display at left of the menu item.
`accel`	accelerator (keystroke) to use to trigger this menu item.
`options`	additional options, for instance 'state = "disable"' to disable the menu at creation.
`active`	do we enable or disable the menu item?
`expr`	an expression to execute corresponding to the menu item call.

Details

Do not use these functions directly. Prefer the corresponding `menuXXX()` functions that will call them if Tk menus or menu items are provided.

Value

`tkMenuAdd()` and `tkMenuAddItem()` return the handle of the newly created menu/menu item invisibly. `tkMenuDel()` and `tkMenuDelItem()` return invisibly `TRUE` if the resource is found and deleted, `FALSE` otherwise. `tkMenuItems()` returns the list of all items in a given menu. `tkMenuInvoke()` returns invisibly `TRUE` if the menu item was invoked, `FALSE` otherwise. `tkMenuStateItem()` returns the new state of the menu. `tkMenuItemCall()` is usually not called directly by the end-user, but rather through a menu. It is a user-visible function so that it is possible to substitute it with a custom function to manage menu item calls differently in a custom GUI, for instance.

Author(s)

Philippe Grosjean

See Also

`menuAdd`

`tkTool` *Conveniently manipulate Tk toolbars*

Description

These functions provide an easy way to create and manipulate Tk toolbars under R. Note that the corresponding `toolXXX()` function also manipulate Tk toolbars the same way, but are capable of manipulating other toolbars as well (not currently, but that could be implemented in the future). One should, thus, preferably use `toolXXX()`!

Usage

```
tkToolAdd(toolbar, side = "top")
tkToolAddItem(toolbar, item, action, image = "", options = "")
tkToolDel(toolbar)
tkToolDelItem(toolbar, item)
tkToolItems(toolbar)
tkToolChangeItem(toolbar, item, action = "", options = "")
tkToolStateItem(toolbar, item, active = TRUE)
tkToolInvoke(toolbar, item)
```

Arguments

`toolbar`	name of a Tk toolbar.
`side`	where to place the toolbar in the window (\"top\", \"bottom\", \"left\", or \"right\")?
`item`	name of a toolbutton.
`action`	action the toolbutton triggers (R code).
`image`	name of a Tk image to display in the toolbutton.
`options`	additional options, for instance 'state = "disable"' to disable the toolbutton at creation.
`active`	do we enable or disable the toolbutton?

Details

Do not use these functions directly. Prefer the corresponding `toolXXX()` functions that will call them if Tk toolbars or toolbutton are provided.

Value

`tkToolAdd()` and `tkToolAddItem()` return the handle of the newly created toolbar or toolbutton invisibly. `tkToolDel()` and `tkToolDelItem()` return invisibly `TRUE` if the resource is found and deleted, `FALSE` otherwise. `tkToolItems()` returns the list of all items in a given toolbar. `tkToolInvoke()` returns invisibly `TRUE` if the toolbutton was invoked, `FALSE` otherwise. `tkToolStateItem()` returns the new state of the toolbutton.

Author(s)

Philippe Grosjean

See Also

`toolAdd`

`tkWin` *Conveniently manipulate Tk windows*

Description

These functions provide an easy way to create and manipulate Tk windows under R. Note that the corresponding `winXXX()` function also manipulate Tk windows the same way, but are capable of manipulating other windows as well (in the future). One should, thus, preferably use `winXXX()`!

Usage

```
tkWinAdd(name = "win1", parent = .TkRoot, title = NULL, pos = NULL,
    bind.delete = TRUE, ...)
tkWinDel(window)
```

Arguments

`name`	name for a new Tk window.
`parent`	parent of this window.
`title`	title of the window.
`pos`	where to place the window. A string like '+XX+YY' where XX is the horizontal position in pixels, and YY is the vertical position. Using negative values place the window relative to the right or bottom side of the screen. Specifying `NULL` (by default) allows for automatic placement of the window.
`bind.delete`	do we automatically bind `tkWinDel()` to the window delete event (strongly advised for correct housekeeping)?
`...`	additional options to pass to the window creator.
`window`	the name of a 'tkguiWin' object.

Details

Do not use these functions directly. Prefer the corresponding `winXXX()` functions that will call them if Tk windows are concerned.

Value

`tkWinAdd()` returns the handle of the newly created window invisibly. `tkMenuDel()` returns invisibly `TRUE` if the window is found and deleted, `FALSE` otherwise.

Author(s)

Philippe Grosjean

See Also

`winAdd`

`Tool` *Conveniently manipulate toolbars, whatever the window*

Description

These functions provide an unifying way of dealing with (simple) toolbars in R. Currently, they support only Tcl/Tk toolbars and toolbuttons, but other graphical toolboxes could be supported too in the future.

Usage

```
toolAdd(toolbar, side = "top")
toolAddItem(toolbar, item, action, image = "", options = "")
toolDel(toolbar)
toolDelItem(toolbar, item)
toolNames()
toolItems(toolbar)
toolType(toolbar, warn = TRUE)
toolInvoke(toolbar, item)
toolChangeItem(toolbar, item, action = "", options = "")
toolStateItem(toolbar, item, active = TRUE)
toolRead(file = "Tools.txt")
toolReadPackage(package, subdir = "gui", file = "Tools.txt")

## S3 method for class 'guiTool'
print(x, ...)
```

Arguments

`toolbar`	name of a toolbar.
`side`	where to place the toolbar in the window (\"top\", \"bottom\", \"left\", or \"right\")?
`item`	name of a toolbar item (a toolbutton).
`action`	action the toolbutton triggers (R code).
`image`	name of an image to display in the toolbutton.
`options`	additional options, for instance "disable" to disable the toolbutton at creation.
`warn`	do we issue a warning if the type of menu is not recognized?
`active`	do we enable or disable the toolbutton?
`file`	a file containing toolbars specifications to read.
`package`	name of a package from where to load toolbars specifications.
`subdir`	subdirectory in the package where the toolbars specifications are stored. By default, it is the "gui" subdirectory.
`x`	an object of class 'guiTool'.
`...`	further arguments (currently not used).

Details

These functions care about creating, deleting and managing custom toolbars. Informations and handles to the various toolbars created with these functions are stored in the .guiTools variable, located in the `SciViews:TempEnv` environment.

Use 'img' resources to load images to display in the toolbuttons.

Value

`toolAdd()`, `toolAddItem()` return the handle to the newly created toolbar or toolbutton invisibly. `toolDel()` and `toolDelItem()` return invisibly `TRUE` if the resource is found and deleted, `FALSE` otherwise. `toolNames()` returns the list of all toolbars registered in .guiTools in the `SciViews:TempEnv` environment. `toolInvoke()` returns invisibly `TRUE` if the toolbutton was invoked, `FALSE` otherwise. `toolRead()` and `toolReadPackage()` return invisibly the list of toolbars that are imported and created.

Author(s)

Philippe Grosjean

See Also

`tkToolAdd`, `imgReadPackage`

Examples

```
## Not run:
## These cannot be run by examples() but should be OK when pasted
## into an interactive R session with the tcltk package loaded
## Run these commands one at a time

winAdd("tt", title = "A Tk window with toolbars", pos ="-40+20")
imgReadPackage("svWidgets")     # Make sure images are loaded
## Create a toolbar and populate it
toolAdd("$Tk.tt/Main")
toolNames()
(toolItems("$Tk.tt/Main"))   # Still nothing in it
toolAddItem("$Tk.tt/Main", "List variables",
    action = "print(ls(envir = .GlobalEnv))", image = "$Tk.butCopy")
toolAddItem("$Tk.tt/Main", "Say yo!", action = "cat('yo!\n')")
toolAddItem("$Tk.tt/Main", "-")
toolAddItem("$Tk.tt/Main", "Search",
    action = "print(search())", image = "$Tk.butPaste")
(toolItems("$Tk.tt/Main"))
## Change state of buttons in the toolbar
toolStateItem("$Tk.tt/Main", "Search", FALSE)
toolStateItem("$Tk.tt/Main", "Search", TRUE)
toolStateItem("$Tk.tt/Main", "Say yo!", FALSE)
toolStateItem("$Tk.tt/Main", "Say yo!", TRUE)
## Invoke a button
toolInvoke("$Tk.tt/Main", "Say yo!")
## Remove a button and add another one (always at the end!)
toolDelItem("$Tk.tt/Main", "Say yo!")
toolAddItem("$Tk.tt/Main", "Say yo! twice", "cat('yo! yo!\n')")
(toolItems("$Tk.tt/Main"))
toolDel("$Tk.tt/Main")
toolNames()
(toolItems("$Tk.tt/Main"))
winDel("tt")

## End(Not run)
```

Win *Manipulate Windows*

Description

R can combine various windows (native, Tk, Gtk, etc.). There could be problems when a GUI uses various kinds of windows together. For instance, it is very difficult to define a modal window that is modal for the whole application. These functions manage windows and ease their clean creation and destruction.

Usage

```
winAdd(name = "win1", type = "tkWin", parent = .TkRoot, title = NULL,
    pos = NULL, bind.delete = TRUE, ...)
winDel(window)
winGet(window)
winNames()

## S3 method for class 'guiWin'
print(x, ...)
```

Arguments

`name`	name for a new window
`type`	type of window to create. Currently, only Tk windows (\"tkWin\") are supported
`parent`	parent of this window
`title`	title of the window
`pos`	where to place the window. A string like '+XX+YY' where XX is the horizontal position in pixels, and YY is the vertical position. Using negative values place the window relative to the right or bottom side of the screen. Specifying `NULL` (by default) allows for automatic placement of the window.
`bind.delete`	do we automatically bind `winDel()` to the windows delete event (strongly advised for correct housekeeping)?
`...`	additional options to pass to the window creator, or the print() method.
`window`	the name of a 'guiWin' object .
`x`	an object of class 'guiWin'.

Details

The list of windows and pointers to their handles are stored in '.guiWins' in the `SciViews:TempEnv` environnement.

Value

`winAdd()` and `winGet()` return the handle to the window (invisibly for `winAdd()`. `winNames()` return the list of all windows registered in .guiWins. `winDel()` returns invisibly `TRUE` if the window is found and deleted, `FALSE` otherwise.

Author(s)

Philippe Grosjean

See Also

`tkWinAdd`, `menuReadPackage`

Examples

```
## Not run:
## These cannot be run by examples() but should be OK when pasted
## into an interactive R session with the tcltk package loaded

## Creating and destroying a Tk window and inspecting the list
winNames()
winAdd("tt", title = "My win", pos ="-40+20")
winNames()
tkwm.deiconify(winGet("tt")) # Standard tcltk functions on the window
winDel("tt")
winNames()

## End(Not run)
```

---

winMenu *Additional winMenuXXX() functions*

Description

A couple of winMenuXXX() functions are defined in the package 'utils' to manipulate custom menus of Rgui (under windows only). Here are some additional ones. Note that you should preferably use the corresponding menuXXX() function defined in this package (they work with windows menus as well as other menus, like Tk.

Usage

```
winMenuChangeItem(menu, item, action, options = "")
winMenuStateItem(menu, item, active = TRUE)
winMenuInvoke(menu, item)
```

Arguments

`menu`	name of a menu.
`item`	name of a menu item.
`action`	action the menu triggers (R code).
`options`	additional options. Only supports "enable" or "disable", currenty.
`active`	do we enable or disable the menu item?

Details

These functions are used only under Windows, and when RgGui.exe is executed.

These functions complement the winMenuXXX() functions in package 'utils'. Do prefer to use the corresponding menuXXX() functions that work with all types of menus (currently, only Windows RGui and Tk, but more could be added in the future).

Author(s)

Philippe Grosjean

See Also

`menuChangeItem`, `menuStateItem`, `menuInvoke`

Index