Is the checkmenu widget enabled ?

Synopsis
isenabled()

Returns
Bool

Description

This tool function indicated whether the checkmenu has been disabled or not.

Example

```
- f := frame();
- m := checkmenu(f, 'Animals', "dog cat");
- m.disable();
- m.isenabled()
F
-
```

1.4.7 combobox - Tool

Package display
Module widgets

A widget for selecting one or more items from a list.
include "combobox.g"

Description

The combobox combines a text entry widget with a popup listbox and a label to provide a handy GUI way to select one or more items from a list. The combobox has an internal data store – a list of strings – and a way to keep track of which of these strings is currently selected. You can think of the listbox as the visual representation of the data store, and the text entry widget as the visual representation of the current selection.

Items in the combobox data store are always strings, so the application must do whatever is necessary to interpret strings as numbers, or file names, or any other application-specific tool. The most common use of the combobox will be for the interactive selection of items by the user in the popup listbox, by using the mouse. The combobox immediately displays these selections in the text extry widget. Users may also edit and type directly into the text entry widget; a carriage return in the text entry widget usually causes the current contents of that widget to be added to the combobox data store (and the listbox) as a new item. The current contents of the text entry widget can be retrieved at any time by a "getentry ()" call. The current listbox selection may be retrieved at any time by a "get('selected')" call. The entry contents may not be the same as the listbox selection if, for example, the user has edited the entry contents but not yet hit return. The application can choose to retrieve the entry contents at any time and, if necessary, add items to the combobox (and optionally make one of these the current selection).

When the user presses the arrow button which causes the listbox to pop up, the listbox grabs the input focus. The user then usually has only two options: clicking the left mouse button on an item in the listbox, which selects that item and puts it into the combobox entry field, or clicking anywhere else which causes the listbox to be dismissed and nothing to happen. If the combobox has been set to display a clear button through the use of the canclearpopup attribute then the user can also click on the clear button. This clears all of the items from the internal data store as well as clearing the entry before dismissing the listbox. If the listbox has scrollbars, the user can manipulate those.

The combobox emits events through the agent returned by the agent member function. Standard events include "select", when the user makes a selection in the combobox or hits return in the entry, causing a new selection, "return", when the user hits return in the entry (which also generates a selection if addonreturn is T) and a "press", when the user presses the button which displays the listbox. The bind member function can be used to bind any x-event to this same agent.

One limitation in glishtk is that it is impossible to query the true size of an entry widget. The best one can do is query the requested size when the widget was constructed. This makes it impossible for the listbox popup to always have the same width as the entry widget in the combobox.

One further caveat is that the indexing into the internal data store through member functions such as get, insert, and delete, is 0-relative. This is to make it identical to the indexing in the glishtk listbox widget which the combobox uses. This means that the first item in the list is at index number 0 and if there are N items in the list the the last item in the list is at index number 'N1'. In addition to being able to index by integer, there are three string indices which are useful: 'active', 'selected', and 'end'. Of these, 'active' and 'end' also can be used in glishtk listbox indexing. The third, 'selected' was added as a natural synonym for 'active'.

The combobox emits events through the agent returned by the "agent" member function.

return *When the user hits return in the combobox entry. The value of this event is the contents of the entry box when the user hit return.*

select *When the user selects something from the popup list box or after a* **return** *if the combobox has addonreturn turned on (the default case).*

clear *When the user presses the "Clear" button on the popup.*

press *When the user presses the arrow button*

user defined *Through the use of the "bind" member function, the application can bind X-events to the entry widget. When a bound event occurs in the entry widget it is emitted through this agent with the name the user chose in the invocation of "bind".*

Example

```
f := frame()
cb := combobox(f,'Colors',"red blue green yellow amber saffron");
print cb.get('selected')
# the string ''red'' is printed out
cb.select(2)
print cb.get('selected')
# the string ''green'' is printed out due to 0-relative indexing
cb.insert('orange', 2)
a := cb.get(0,'end')
# a now contains "red blue orange green yellow amber saffron"
# The selection is maintained at green, which now has an index of 3
```

combobox - Constructor

Package	**display**
Module	**widgets**
Tool	**combobox**

Construct a combobox

Synopsis

combobox(parent, labeltext, items, addonreturn, entrydisabled, canclearpopup, autoinsertorder, disabled, borderwidth, exportselection, vscrollbar, hscrollbar, labelwidth, labelfont, labelrelief, labeljustify, labelbackground, labelforeground, labelanchor, entrywidth, entryfont, entryrelief, entryjustify, entrybackground, entryforeground, entryanchor, arrowbutton, arrowbackground, arrowforeground, listboxheight, help, widgetset)

Most of the defaulted arguments provide fine control over the appearance of the various parts of the combobox. Most users will never need to use these arguments. The arguments most likely be changed by users are placed early in the argument list.

Arguments

parent	*a frame to hold the combobox* *Allowed: glishtk frame*
labeltext	*a label for the combobox* *Allowed: string* *Default: 'label'*
items	*The initial contents of the combobox data. The first item will be selected. If F the contents will be empty* *Allowed: an array of strings* *Default: F*
addonreturn	*Should a return in the entry widget automatically add the entry value to the contents?* *Allowed: T or F* *Default: T*
entrydisabled	*When the entry is disabled, the user can not edit in the entry, the combobox otherwise functions normally.* *Allowed: T or F* *Default: F*
canclearpopup	*When this is true, the listbox comes with a "Clear" button which, when pressed, clears the contents.* *Allowed: T or F* *Default: F*
autoinsertorder	*Determines where each new item is inserted if no specific location is given.* *Allowed: 'tail','head'* *Default: 'tail'*
disabled	*Disable the combobox, turns off all graphical interaction.* *Allowed: T or F* *Default: F*
borderwidth	*The borderwidth of the outer combobox frame.* *Allowed: any integer* *Default: 2*
exportselection	*exportselection value for entry.* *Allowed: T or F* *Default: T*
vscrollbar	*The type of vertical scrollbar that the listbox has.* *Allowed: 'ondemand','always','none'* *Default: 'ondemand'*
hscrollbar	*The type of horizontal scrollbar that the listbox and the entry have.* *Allowed: 'ondemand','always','none'* *Default: 'none'*
labelwidth	*The width, in characters, of the label.* *Allowed: any integer* *Default: 0*
labelfont	*The font used in the label.* *Allowed: any X font.*
labelrelief	*The relief of the label.* *Allowed: any glishtk relief* *Default: 'flat'*
labeljustify	*The text justification of the label.* *Allowed: any glishtk text justificiation* *Default: 'left'*

Arguments

labelbackground	*The background color of the label.* *Allowed: any X color.* *Default: 'lightgrey'*
labelforeground	*The foreground color of the label.* *Allowed: any X color.* *Default: 'black'*
labelanchor	*The text anchor point within the label.* *Allowed: any glishtk anchor.* *Default: 'c'*
entrywidth	*The width, in characters, of the entry.* *Allowed: any integer* *Default: 30*
entryfont	*The font used in the entry.* *Allowed: any X font.*
entryrelief	*The relief of the entry.* *Allowed: any glishtk relief* *Default: 'sunken'*
entryjustify	*The text justification of the entry.* *Allowed: any glishtk text justificiation* *Default: 'left'*
entrybackground	*The background color of the entry.* *Allowed: any X color.* *Default: 'lightgrey'*
entryforeground	*The foreground color of the entry.* *Allowed: any X color.* *Default: 'black'*
entryanchor	*The text anchor point within the entry.* *Allowed: any glishtk anchor.* *Default: 'c'*
arrowbutton	*The file containing the bitmap to use for the arrow button.* *Allowed: any X bitmap file.* *Default: 'downarrow.xbm'*
arrowbackground	*The background color of the arrow button.* *Allowed: any X color.* *Default: 'lightgrey'*
arrowforeground	*The foreground color of the arrow button.* *Allowed: any X color.* *Default: 'black'*
listboxheight	*The height of the listbox when a horizontal scrollbar is present.* *Allowed: any integer* *Default: 6*
help	*Short popup help text* *Allowed: String*
widgetset	*Widget set to use* *Allowed: A widgetserver closure tool* *Default: dws*

Returns
a combobox tool with associated functions.

---

combobox.agent - Function

Package	**display**
Module	**widgets**
Tool	**combobox**

Get a reference to the agent which emits events from this combobox.

Synopsis
agent()

Returns
a glish agent

Description
This combobox agent emits a "return" when the user hits return in the entry, a "press" the user presses the arrow button, a "select" when then makes a selection in the listbox, and "clear" when the user presses the clear button on the listbox (which is only present if canclearpopup is T. A "select" event also follows a "return" event if addonreturn is T.

Example

```
f := frame()
cb := combobox(f,'colors')
whenever cb.agent()->* do {
    print $name, ' event seen, value is : ', $value
        }
```

---

combobox.pixelwidth - Function

Package	**display**
Module	**widgets**
Tool	**combobox**

The true width, in pixels, of the outer combobox frame.

Synopsis
pixelwidth()

Returns
an integer

---

combobox.pixelheight - Function

Package	**display**
Module	**widgets**
Tool	**combobox**

The true height, in pixels, of the outer combobox frame.

Synopsis
pixelheight()

Returns
an integer

---

combobox.get - Function

Package	**display**
Module	**widgets**
Tool	**combobox**

Get an item or range of items from the combobox.

Synopsis
get(first, last)

Arguments

first	*The first item to get* *Allowed: Any valid item number from 0 to the end plus 'selected' and 'end'*
last	*The last item to get, if F just get the first* *Allowed: Any index ¿= first or F* *Default: F*

Returns
A vector of strings

Description
As with the glishtk listbox, the index numbers here are 0-relative (i.e. the first item is item number 0). "selected" translates to the index of the current selection, "end" translates to the last item in the combobox. get() returns a fail if the index is out of range or otherwise does not exist (e.g. there is no current selection).

---

combobox.selection - Function

Package	**display**
Module	**widgets**
Tool	**combobox**

Retrieve the index number of the current selection.

Synopsis
selection()

Returns
An integer

Description
Returns a -1 if there is no current selection.

---

combobox.select - Function

Package	**display**
Module	**widgets**
Tool	**combobox**

Select the indicated item by index number and place it in the entry.

Synopsis
select(whichitem)

Arguments

whichitem	*The item to select* *Allowed: Any valid item number from 0 to the end plus 'selected' and 'end'*

Returns
T

Description
This returns a fail if there is no valid selection.

---

combobox.getentry - Function

Package	**display**
Module	**widgets**
Tool	**combobox**

Get the contents of the entry.

Synopsis
getentry()

Returns
A string

---

combobox.addonreturn - Function

Package	**display**
Module	**widgets**
Tool	**combobox**

Sets the behavior when the user presses return in the entry.

Synopsis
addonreturn(addonreturn)

Arguments

addonreturn	*The requested behavior* *Allowed: T or F*

Description
When the argument is T, a return automatically adds the contents of the entry to the combobox internal data, when it is F, nothing happens. Typically, when this attribute is F the application will wait for return events from the combobox and examine the value of the entry before deciding if it should be added to the internal combobox data or not. The default behavior is to always add values on return, or T.

Example

```
f := frame()
cb := combobox(f,'colors',"red blue green yellow amber saffron")
cb.addonreturn(F)
whenever cb.agent()->return do {
    newitem := $value
    if (newitem != cb.get('selected')) {
        # only add a new item if it isn't what is currently
        # selected
        cb.insert(newitem, select=T)
        # when added, insert it via current autoinsertorder
        # and select it
    }
}
```

---

combobox.canclearpopup - Function

Package	**display**
Module	**widgets**
Tool	**combobox**

Is there a clear button on the listbox popup.

Synopsis
canclearpopup(tOrF)

Arguments

tOrF	*Is there a clear button* *Allowed: T or F*

Description
If this is true, then a clear button is present next to the listbox popup which, when pressed, clears the contents of the combobox (deleting them all and clearing the entry). Otherwise, there is no graphical way to clear the contents of the combobox. The default behavior is to have a clear button, or F.

---

combobox.borderwidth - Function

Package	**display**
Module	**widgets**
Tool	**combobox**

Change the borderwidth of the outer combobox frame.

Synopsis
borderwidth(newwidth)

Arguments

newwidth	*The new width* *Allowed: any positive integer.*

---

combobox.exportselection - Function

Package	**display**
Module	**widgets**
Tool	**combobox**

Change the export selection property of the entry.

Synopsis
exportselection(tOrF)

Arguments

tOrF	*The new exportselection value* *Allowed: T or F.*

---

combobox.disabled - Function

Package	**display**
Module	**widgets**
Tool	**combobox**

Disable (T) or enable (F) the entire combobox.

Synopsis
disabled(tOrF)

Arguments

tOrF	*The new state of the combobox* *Allowed: T or F.*

Description
When the combobox is disabled, the user can not interact with it graphically although all of the functions work as always. The state of the entry is remember so that if the entry was disabled when the combobox was disabled, it will remain disabled when the combobox is enabled (via a F argument to this function).

---

combobox.delete - Function

Package	**display**
Module	**widgets**
Tool	**combobox**

Delete an item or range of items from the combobox.

Synopsis
delete(first, last)

Arguments

first	*The first item to delete*	
	Allowed:	*Any valid item number from 0 to the end plus 'selected' and 'end'*
last	*The last item to delete, if F just get the first*	
	Allowed:	*Any index ¿= first or F*
	Default:	*F*

Description
This deletes these items from the list. If the current selection falls within the specified range, the first item before the range is selected or the first item after the range is selected if the range includes item 0. If the entire contents are deleted, there is no current selection after this function has returned. As with the glishtk listbox, the index numbers here are 0-relative (i.e. the first item is item number 0). "selected" translates to the index of the current selection, "end" translates to the last item in the combobox. delete() returns a fail if the index is out of range or otherwise does not exist (e.g. there is no current selection) and the contents of the combobox will remain unchanged in that case.

Example

```
f := frame();
        cb := combobox(f,'colors',"red blue green yellow amber saffron")
        # delete everything from the current selection to the end
        # the new selection is whatever immediately preceeds the current
        # selection or nothing, if 'selected' is item 0.
        cb.delete('selected','end')
```

---

combobox.insert - Function

Package	**display**
Module	**widgets**
Tool	**combobox**

Insert an item into the combobox.

Synopsis
insert(newitem, index, select)

Arguments

newitem	*The thing to insert* *Allowed: A string*
index	*Its new location* *Allowed: Any index ¡= largest index + 1* *Default: F*
select	*Select this item* *Allowed: T or F* *Default: F*

Returns
T

Description
If index is F then the current autoinsertorder determines which and of the list of items this item is added at. If an index is specified, all of the existing items from that index to the end are moved to the next location (i.e. an insert at location 5 results in the indexes of 5 to the end being now 6 to the new end). This may cause the index of the current selection to change. As with the glishtk listbox, the index numbers here are 0-relative (i.e. the first item is item number 0). "selected" translates to the index of the current selection, "end" translates to the last item in the combobox. insert() returns a fail if the index is out of range (more than 1 larger than the current largest index). If select is T, this item will be selected after insertion.

---

combobox.insertentry - Function

Package	**display**
Module	**widgets**
Tool	**combobox**

Replace the contents of the entry.

Synopsis

insertentry(newentry)

Arguments

newentry	*The new contents of the entry* *Allowed: A string*

Description
This new entry is not selected and it does not appear in the combobox popup. In other words, this is just as if the user had typed this string and not yet hit return in the entry.

---

combobox.vscrollbar - Function

Package	**display**
Module	**widgets**
Tool	**combobox**

Change the vertical scrollbar behavior of the combobox popup.

Synopsis
vscrollbar(vscrollbar)

Arguments

vscrollbar	*The new vertical scrollbar behavior* *Allowed: 'ondemand', 'always', 'none'*

Description
When 'ondemand' is requested, the vertical scrollbar only appears when the number of items is larger than listboxheight elements. When 'always' is requested, the listbox always has a scrollbar. When 'none' is requested, the listbox never has a scrollbar and it is always high enough to show all of the items in the combobox.

---

combobox.hscrollbar - Function

Package	**display**
Module	**widgets**
Tool	**combobox**

Change the horizontal scrollbar behavior of the entry and the combobox popup.

Synopsis
hscrollbar(hscrollbar)

Arguments

hscrollbar	*The new horizontal scrollbar behavior* *Allowed: 'ondemand', 'always', 'none'*

Description
The entry and listbox popup share the same horizontal scrollbar behavior. When one has a horizontal scrollbar, the other does as well. When 'ondemand' is requested, the horizontal scrollbar only appears when the number longest item is longer than entrywidth elements. When 'always' is requested, the horizontal scrollbar is always present. When 'none' is requested, there is no horizontal scrollbar and the listbox is always wide enough to show all of the items in the combobox.

---

combobox.autoinserorder - Function

Package	**display**
Module	**widgets**
Tool	**combobox**

Change the automatic insertion behavior.

Synopsis
autoinserorder(insertorder)

Arguments

insertorder	*The new automatic insertion behavior* *Allowed: 'tail', 'head'*

Description
This determines how new items are added to the combobox when no specific index location is specified - either through the use of the insert function or through the users actions in the entry field. If this is 'tail', the new item is added to the end of the list, if this is 'head', the new item is added to the head of the list.

combobox.bind - Function

Package	**display**
Module	**widgets**
Tool	**combobox**

Bind an X event to the combobox entry.

Synopsis
bind(xevent, eventname)

Arguments

xevent	*The X event to be bound.* *Allowed: An X event in glishtk syntax*
eventname	*The name which will be associated with this event.* *Allowed: A string*

Description
This event will be bound to the combobox entry. Events having the name eventname will be emitted from the agent returned by the agent member function each time this X event occurs within the combobox entry.

Example

```
cb.bind('<KeyPress>','keypress')
whenever cb.agent()->keypress do {
   print 'Key pressed in combobox entry : ', $value;
}
```

combobox.cursor - Function

Package	**display**
Module	**widgets**
Tool	**combobox**

Change the default cursor within the combobox.

Synopsis
cursor(xcursor)

Arguments

xcursor	*The new default cursor.* *Allowed: An X cursor*

---

combobox.popupcursor - Function

Package	**display**
Module	**widgets**
Tool	**combobox**

Change the default cursor within the combobox popup.

Synopsis
popupcursor(xcursor)

Arguments

xcursor	*The new default cursor in the popup.* *Allowed: An X cursor*

---

combobox.labeltext - Function

Package	**display**
Module	**widgets**
Tool	**combobox**

Change the text of the combobox label.

Synopsis
labeltext(newlabel)

Arguments

newlabel	*The new label text.* *Allowed: A string.*

---

combobox.labelwidth - Function

Package	**display**
Module	**widgets**
Tool	**combobox**

Change the width of the combobox label.

Synopsis

labelwidth(newwidth)

Arguments

newwidth	*The new label width.* *Allowed: An integer.*

---

combobox.labelfont - Function

Package	**display**
Module	**widgets**
Tool	**combobox**

Change the font of the combobox label.

Synopsis

labelfont(newfont)

Arguments

newfont	*The new label font.* *Allowed: An X font.*

---

combobox.labelrelief - Function

Package	**display**
Module	**widgets**
Tool	**combobox**

Change the relief of the combobox label.

Synopsis

labelrelief(newrelief)

Arguments

newrelief	*The new label relief.* *Allowed: A glishtk relief string.*

---

combobox.labeljustify - Function

Package	**display**
Module	**widgets**
Tool	**combobox**

Change the text justification within the combobox label.

Synopsis

labeljustify(newjustify)

Arguments

newjustify	*The new label text justification.* *Allowed: A glishtk justify string.*

---

combobox.labelforeground - Function

Package	**display**
Module	**widgets**
Tool	**combobox**

Change the foreground color of the combobox label.

Synopsis
labelforeground(newforeground)

Arguments

newforeground	*The new label foreground color.* *Allowed: An X color.*

---

combobox.labelbackground - Function

Package	**display**
Module	**widgets**
Tool	**combobox**

Change the background color of the combobox label.

Synopsis
labelbackground(newbackground)

Arguments

newbackground	*The new label background color.* *Allowed: An X color.*

---

combobox.labelanchor - Function

Package	**display**
Module	**widgets**
Tool	**combobox**

Change the anchor attribute of the combobox label.

Synopsis
labelanchor(newanchor)

Arguments

newanchor	*The new label anchor attribute.* *Allowed: A glishtk anchor string.*

---

combobox.entrydisabled - Function

Package	**display**
Module	**widgets**
Tool	**combobox**

Disable or enable the entry.

Synopsis

entrydisabled(tOrF)

Arguments

tOrF	*The state of the entry.* *Allowed: T or F.*

Description

When the entry is disabled (T here), the user can not type into the entry part of the combobox. This makes it impossible for the user to add to the contents of the combobox graphically. The combobox functions still behave as before so an application is free to do whatever it wants to do to the combobox. This allows an application to present an unchangable set of combobox options to the user. The user can only choose from the existing set of items. They can not add to them. If the application programmer has choosen to provide a clear button in the listbox, however, it would then be possible for the user to clear the contents of the combobox.

---

combobox.entrywidth - Function

Package	**display**
Module	**widgets**
Tool	**combobox**

Change the width of the combobox entry.

Synopsis
entrywidth(newwidth)

Arguments

newwidth	*The new entry width.* *Allowed: An integer.*

Description
Note that this will affect the width of the listbox popup when no horizontal scrollbar is present.

---

combobox.entryfont - Function

Package	**display**
Module	**widgets**
Tool	**combobox**

Change the font of the combobox entry and popup.

Synopsis
entryfont(newfont)

Arguments

newfont	*The new entry font.* *Allowed: An X font.*

---

combobox.entryrelief - Function

Package	**display**
Module	**widgets**
Tool	**combobox**

Change the relief of the combobox entry and popup.

Synopsis
entryrelief(newrelief)

Arguments

newrelief	*The new entry relief.* *Allowed: A glishtk relief string.*

---

combobox.entryjustify - Function

Package	**display**
Module	**widgets**
Tool	**combobox**

Change the text justification within the combobox entry and popup.

Synopsis

entryjustify(newjustify)

Arguments

newjustify	*The new entry text justification.* *Allowed: A glishtk justify string.*

---

combobox.entryforeground - Function

Package	**display**
Module	**widgets**
Tool	**combobox**

Change the foreground color of the combobox entry and popup.

Synopsis

entryforeground(newforeground)

Arguments

newforeground	*The new entry foreground color.* *Allowed: An X color.*

---

combobox.entrybackground - Function

Package	**display**
Module	**widgets**
Tool	**combobox**

Change the background color of the combobox entry and popup.

Synopsis
entrybackground(newbackground)

Arguments

newbackground	*The new entry background color.* *Allowed: An X color.*

---

combobox.arrowbutton - Function

Package	**display**
Module	**widgets**
Tool	**combobox**

Change the bitmap which is the arrow button.

Synopsis
arrowbutton(newbutton)

Arguments

newbutton	*The new arrow button bitmap.* *Allowed: The name of a file containing an X bitmap.*

---

combobox.arrowbackground - Function

Package	**display**
Module	**widgets**
Tool	**combobox**

Change the background color of the arrow button.

Synopsis
arrowbackground(newbackground)

Arguments

newbackground	*The new arrow button background color.* *Allowed: An X color.*

---

combobox.arrowforeground - Function

Package	**display**
Module	**widgets**
Tool	**combobox**

Change the foreground color of the arrow button.

Synopsis
arrowforeground(newforeground)

Arguments

newforeground	*The new arrow button foreground color.* *Allowed: An X color.*

---

combobox.listboxheight - Function

Package	**display**
Module	**widgets**
Tool	**combobox**

Change the height of the listbox.

Synopsis
listboxheight(newheight)