activate()

Description
Remove the progress meter window from the screen. If the window is already inactive, the function is a no-op.

---

progress.deactivate - Function

Package	**display**
Module	**guiutils**
Tool	**progress**

Return the window to the screen

Synopsis
deactivate()

Description
If the window is not currently active on the screen, this function returns it, even if the window has previously hit its maximum. If the window is already active, this function is a no-op.

1.1.16 displaylist - Tool

Package	**display**
Module	**guiutils**

Maintain an editable list of items
include "displaylist.g"

Description

A display list is an editable list of items. An item may either be in the "draw" list, or it may be in the "clipboard". An item moves back and forth via `cut`, `copy`, and `paste` operations.
This class is not intended to be used by general users, rather it is intended to be used in the constructions of GUI's.

Example
See global function, `displaylisttest()`

displaylist - Constructor

Package	**display**
Module	**guiutils**
Tool	**displaylist**

Create a displaylist tool

Synopsis
displaylist(parentframe, formatfunction)

Description
Create a list tool ready to start adding values to. Unless a frame is given to the constructor, the default is to not have a GUI representation of the displaylist. Generally you should provide a function to turn inserted values into strings to display in the list. By default, `as_string()` will be used, which is rarely what you will want.

Arguments

parentframe	frame to use for gui Allowed:frame or F Default: F (no frame)
formatfunction	turn list values into string for display Allowed:function

Returns
displaylist tool

---

displaylist.add - Function

Package	**display**
Module	**guiutils**
Tool	**displaylist**

add a value to the display list

Synopsis
add(value)

Description
This function adds a new value to the draw list. Note that you may add any type of value to the display list, although generally you will add record types

Arguments

value	item to add to the draw list Allowed:any

Returns
number of entries in the draw list

---

displaylist.get - Function

Package	**display**
Module	**guiutils**
Tool	**displaylist**

get an item from the displaylist

Synopsis
get(num, fromdrawlist)

Description
Use an index (1 relative) to get an item from either the draw list or the clipboard (by default the draw list). You can determine how many items there are by calling `ndrawlist()` or `nclipboard()`.

Arguments

num	index Allowed:integer
fromdrawlist	get from drawlist? Allowed:T

Returns
the item or fail if index invalid

---

displaylist.set - Function

Package	**display**
Module	**guiutils**
Tool	**displaylist**

change or add an item in displaylist

Synopsis
set(num, value)

Description
Change or an existing item in the draw list (only) or add a new one. An existing item is changed if the index is in the range 1–`ndrawlist()`, otherwise a new item is added. (The index should only be one larger than the current length of the drawlist).

Arguments

num	index Allowed:integer
value	item to change or add in the draw list Allowed:any

Returns
the new length of the draw list or fail if index invalid

---

displaylist.clear - Function

Package	**display**
Module	**guiutils**
Tool	**displaylist**

clear the draw list and clipboard

Synopsis
clear()

Description
Empties all items in the draw list and clipboard.

---

displaylist.cleardrawlist - Function

Package	**display**
Module	**guiutils**
Tool	**displaylist**

clear the draw list

Synopsis
cleardrawlist()

Description
Empties all items in the draw list.

---

displaylist.clearclipboard - Function

Package	**display**
Module	**guiutils**
Tool	**displaylist**

clear the clipboard

Synopsis
clearclipboard()

Description
Empties all items in the clipboard.

---

displaylist.ndrawlist - Function

Package	**display**

Module **guiutils**
Tool **displaylist**

how many items are in the draw list

Synopsis
ndrawlist()

Description
Returns the number of items currently in the draw list.

Returns
integer

---

displaylist.nclipboard - Function

Package **display**
Module **guiutils**
Tool **displaylist**

how many items are in the clipboard

Synopsis
nclipboard()

Description
Returns the number of items currently in the clipboard.

Returns
integer

---

displaylist.cut - Function

Package **display**
Module **guiutils**
Tool **displaylist**

move items from the draw list to the clipboard

Synopsis
cut(drawnums)

Description
Cut is used to move items (desiganted as an array of integers) from the draw list to the clipboard.

Arguments

drawnums	index array Allowed:integer

Returns
the length of the drawlist or fail

---

displaylist.copy - Function

Package	**display**
Module	**guiutils**
Tool	**displaylist**

copy items from the draw list to the clipboard

Synopsis
copy(drawnums)

Description
Copy is used to to copy items (desiganted as an array of integers) from the draw list to the clipboard.

Arguments

drawnums	index array Allowed:integer

Returns
the length of the drawlist or fail

---

displaylist.paste - Function

Package	**display**
Module	**guiutils**
Tool	**displaylist**

copy items from clipboard to the drawlist

Synopsis
paste()

Description
Paste is used to copy items from the clipboard to the drawlist. At present all items from the clipboard are copied to the end of the drawlist. We should probably at least allow pasting into the middle of the draw list.

Returns
the length of the drawlist or fail

---

displaylist.compact - Function

Package	**display**
Module	**guiutils**
Tool	**displaylist**

restore wasted space in tool

Synopsis
compact()

Description
After many editing operations there can be wasted space inside the tool. Compact reclaims this space. No visibile change occurs in the state of the tool.

---

displaylist.apply - Function

Package	**display**
Module	**guiutils**
Tool	**displaylist**

call a function on items

Synopsis
apply(fun, dodraw)

Description
Apply applies a function to every item in the draw list (default) or clipboard. The function must take a single argument (or all the other arguments must default), *i.e.*, the item it is being passed.

Arguments

fun	the function we are applying to items Allowed:function
dodraw	apply to items in draw list or clipboard Allowed:boolean Default: T (draw)

Returns
the length of the drawlist or or clipboard or fail

---

displaylist.lastchange - Function

Package	**display**
Module	**guiutils**
Tool	**displaylist**

return a number that specifies the state of the displaylist

Synopsis
lastchange()

Description

It is sometimes useful to know whether or not the displaylist has changed since the last time you have looked at it (given that the displaylist will often be under user control through some GUI). If the number returned by this function has changed since the last time you called it, it has. The number returned is always non-negative, so you can initialize your cache with a negative number.

Returns
integer

---

displaylist.gui - Function

Package	**display**
Module	**guiutils**
Tool	**displaylist**

show the displaylist in a GUI

Synopsis
gui(parentframe)

Description
Create a GUI for the displaylist using a supplied frame. The clipboard appears in a listbox on the left, the draw list in a listbox on the right, and various buttons between them to control cutting, pasting, *etc.* Items are selected by clicking and dragging in the listboxes.

Arguments

parentframe	parent frame to write into Allowed:frame

Returns
T or fail

---

displaylist.nogui - Function

Package	**display**
Module	**guiutils**
Tool	**displaylist**

turn off GUI view of displaylist

Synopsis
nogui()

Description
Turns off the GUI representation of this display list, but does not clear the actual items in the draw list and clipboard. Note that this function sets the parentframe of `gui()` to F.

---

displaylist.save - Function

Package	**display**
Module	**guiutils**
Tool	**displaylist**

save the displaylist into a file

Synopsis
save(filename)

Description
This function saves the displaylist (including the clipboard), into a file. At present, the file format is just a Glish value, but this is subject to change. If the file already exists, it is overwritten. A fail is returned if the file cannot be written to (probably a read-only or non-existent directory).

Arguments

filename	file to write into Allowed:string

Returns
T or fail

---

displaylist.restore - Function

Package	**display**
Module	**guiutils**
Tool	**displaylist**

restore the displaylist from a file

Synopsis
restore(filename)

Description
This function restores the displaylist from a file. The current contents are removed first. At present there is no way to merge from a file, but this could be added if desired.

Arguments

filename	file to read from Allowed:string

Returns
T or fail on read failure

---

displaylist.setselectcallback - Function

Package	**display**
Module	**guiutils**
Tool	**displaylist**

call a function when an item is selected

Synopsis
setselectcallback(callback)

Description
This sets a callback function to be called whenever an item is selected in the draw list of the GUI. The function must take one argument - the selected item.

Arguments

callback	function to call on item Allowed:function

Returns
T or fail

---

displaylist.setchangecallback - Function

Package	**display**
Module	**guiutils**
Tool	**displaylist**

call a function when the list changes

Synopsis
setchangecallback(callback)

Description
This sets a callback function to be called when the list is first changed, at which point the callback is unset. The idea is that the user somehow "saves" the drawlist, at which point he wants to be notified when the list has changed. When he has saved it again, he resets the callback.

Arguments

callback	function to call on item Allowed:function

Returns
T or fail

---

displaylist.setrefreshcallback - Function

Package	**display**
Module	**guiutils**
Tool	**displaylist**

call function on cut and paste

Synopsis
setrefreshcallback(callback)

Description
Set a function to be called whenever there is a cut or paste. Arguably, the function should be called at other times (e.g., copy), or at least there should be additional callback functions. This function must take no arguments.

Arguments

callback	function to call on item Allowed:function

Returns
T or fail

---

displaylist.done - Function

Package	**display**
Module	**guiutils**
Tool	**displaylist**

release/destory this tool

Synopsis
done()

Description
Destroy all resources (GUI and the item lists) allocated by this tool.

1.1.17 displaylisttest - Function

Package	**display**
Module	**guiutils**

Test/demonstration of the displaylist class

Synopsis
displaylisttest()

Description
Combination test/demonstration of displaylist.

1.1.18 popuphelp - Function

Package display
Module guiutils

add popup help to a GUI

Synopsis
popuphelp(fr, txt, hlp, combi, width)
include "popuphelp.g"

Description
`popuphelp` Is a function which adds the code necessary to add "popup" help to an agent. In particular, when your cursor "lingers" or "hovers" over that agent (e.g. a button) for more than about half a second, the help will popup, and go away when the cursor leaves the agent or when a button is pushed, etc.
Help text comes in two flavours: short and long. If only one of them is available, they will both be identical. Short help is displayed on one line if it is not longer than the specified width (60 by default), else on several lines. If no explicit long help is present, the shorthelp is truncated at a new line or at a period, and will never be longer than the specified width. If the longhelp differs from the shorthelp (either by truncation or otherwise), the shorthelp is extended with ".." at the end of the displayed short text.
Short help is displayed by default when the cursors hovers above the agent.
Long help is tied to the right mouse button by default.
Popup help can, dynamically, be tied by setting the 'system.help.type' to 0 (no popup), 1(short and long), 2(long only with MB3), 3(long only in hover).
A "popupmenu" function is also available to add a "?" button to your GUI which let the user choose the context popup type action. If a popupmenu is generated, the default can be set with the aipsrc variable "help.popup.type" (possible values: 'none', 'both', 'mb3long' and 'hoverlong'), or by the programmer.
For most popup help items you merely specify a text string that is the help text. For a listbox, you can supply a callback function that takes two arguments and returns a string. The two supplied arguments are:

1. the text of the listbox item the cursor is near; and
2. the Y-index (0 relative!) of the cursor position in the listbox.

It should be noted that timing errors could crash the system (or leave popuphelp floating around) if an agent that supports popuphelp is just zeroed (e.g. by frameagent :=F). A function 'popupremove(agent)' exists to do a proper zeroing.

Arguments

fr	agent to which popup text should be added Allowed:record or agent
txt	popup text, or, for listbox, possible callback routine Allowed:string or function as described above Default: F
hlp	optional separate short help text, or,for listbox, possible callback routine Allowed:string or function as described above Default: F
combi	specifies if short text should precede long text Allowed:bool Default: F
width	specifies maximum length of one shorthelp line Allowed:integer Default: 60

Returns
T (or fail)

Example

```
include 'popuphelp.g';
private := [=];
private.f := frame(height=100);
private.b := button(private.f);
popupmenu(private.f);  # to add a ? menu button (if wanted)
popuphelp(private.b, 'Press the button, and make this a long text');
private.l := listbox(private.f);
private.l->insert('AAA'); private.l->insert('BBB');
private.l.helpfct := function(name, index) {
    # Only need name here.
    return spaste('Your cursor is near ', name);
}
popuphelp(private.l, private.l.helpfct);
```

1.1.19 popupremove - Function

Package display
Module guiutils

remove popup help from a GUI

Synopsis
popupremove(ag, mxlevels)
include "popuphelp.g"

Description
`popupremove` Is a function which deletes any popuphelp (if present) from the specified agent, and sets the agent to False after having done it. Using it is necessary if, e.g., you want to zero the agent in a whenever: timing problems could crash the system if popuphelp is present and lingering somewhere.
The function works recursively, if fields in the argument are records.

Arguments

ag	agent which should be removed (possibly recursively) Allowed:agent Default: F
mxlevels	max. recursion level Allowed:Int Default: 8

Returns
T

Example

```
include 'popuphelp.g';
private := [=];
private.f := frame(height=100);
private.f.a := frame(private.f);
private.f.a.b := button(private.f.a);
popupmenu(private.f);  # to add a ? menu button (if wanted)
popuphelp(private.f.a.b, 'Press the button, and make this a long text');
private.l := listbox(private.f);
```

```
private.l->insert('AAA'); private.l->insert('BBB');
private.l.helpfct := function(name, index) {
    # Only need name here.
    return spaste('Your cursor is near ', name);
}
popuphelp(private.l, private.l.helpfct);
whenever private.f.a.b->press do {
    popupremove(private.f.a);
}
```

1.1.20 popupmenu - Function

Package **display**
Module **guiutils**

add a popup type selection help menu to a GUI

Synopsis
popupmenu(agent, deflt, relief)
include "popuphelp.g"

Description
`popupmenu` Is a function which adds to an agent a "?" button with popup selection options of "none", "both", "mb3" or "hover". By specifying a deflt argument of 0, 1, 2 or 3 you can select one of them as the default (default both assumed).

Arguments

agent	agent to which menu button should be added Allowed:agent
deflt	default popup useage type Allowed:integer Default: F
relief	button relief Allowed:relief type string Default: flat

Returns
T (or fail)

Example
See “popuphelp”.

1.1.21 addpopuphelp - Function

Package display
Module guiutils

add popup help to a GUI

Synopsis
addpopuphelp(agents, maxlevels)
include ”popuphelp.g”

Description
`addpopuphelp` Is a function which recurses through a data structure (typically this will be the “private” variable of your closure tool, and for every agent that it finds that has a field named “shorthelp”, it will add the code necessary to add “popup” help to that agent.
It is advised to use the “popuphelp” function directly in stead.
Sometimes record/agent structures are set up where a subrecord is in fact a reference to a parent record. Since this might cause infinite recurses, a `maxlevels` parameter is used to prevent such infinite loops. It is set to 4 by default, which should generally suffice, but very deep structures may require this number to be raised.

Arguments

agents	structure that holds the agents Allowed:record or agent
maxlevels	maximum number of levels to recurse Allowed:4

Returns
T (or fail)

Example