dirName	Directory name Allowed:Any directory name Default: Current directory
restrictions	Whether to show tables or files Allowed:Two element boolean record Default: [T,T]
title	Title to put on Window Allowed:Any String Default: AIPS++ chooser
wait	Flag for whether to wait for result or not Allowed:T or F Default: T
writeOK	Flag for allowing writing of files Allowed:T or F Default: T
multi	Flag for allowing multiple choices Allowed:T or F Default: F
access	String specifying required access permissions of files Allowed:any combination of r,w, and x Default: r

Returns
An agent, sends a returns event when OK button is pressed

1.1.8 guiframework - Tool

Package display
Module guiutils

Standard framework for Glish based AIPS++Graphical User Interfaces (GUI's)
include "guiframework.g"

Description
The guiframework (gf) provides a framework for constructing consistent graphical user interfaces. It handles the overhead of the standard set of widgets that most GUI's should have, menubars, command buttons, status lines. Customization of the menu bar is fairly straight forward with the use of Glish records.
A menu record consists of

```
menu_name::text              #! Text appearing on the Menu Button
menu_name.menu_item
```

```
menu_name.menu_item.text      #! Text appearing on the Menu Item
menu_name.menu_item.relief
menu_name.menu_item.action
menu_name.menu_item.type
menu_name.menu_item.state
```

An action record consists of

```
action_name.text
action_name.action
```

The functions, addactions and addmenus, expect the action and menu records to live off a root menu or action tree (i.e. menu_root.menu_name.menu_item). Any valid glishTk button attribute maybe set for each menu or action item. The following member functions illustrate how to construct the menu/action records.

default_filemenu make a simple menu,

default_options make a menus with cascading menus

defaultmenus put together a menus record

defaulthelp put together a help record

default_actions create an actions record

```
self.default_filemenu := function()
{  wider public;
   file := [=];
   file::text := 'File';
   file.open := [=];
   file.open.text := 'Open...';
   file.open.relief := 'flat';
   file.open.action := function(){
               return filechooser(title='AIPS++ Filechooser -- Open File');}
   file.save := [=];
   file.save.text := 'Save...'
   file.save.relief := 'flat';
   file.save.action := function(){
               return filechooser(title='AIPS++ Filechooser -- Save File');}
   file.blank := [=];
   file.blank.text := '';
   file.blank.relief :='flat';
   file.blank.disable :=T;

   file.bug := [=];
   file.bug.text :='Report Bug...';
```

```
   file.bug.relief :='flat';
   file.bug.action := bug;

   file.blank1 := [=];
   file.blank1.text := '';
   file.blank1.relief :='flat';
   file.blank1.disable :=T;

   file.close := [=];
   file.close.text := 'Close';
   file.close.relief := 'flat';
   file.close.action := public.dismiss;

   return ref file ;
}

self.default_options := function()
{
   options := [=];
   options::text := 'Options'
   options.browser := [=];
   options.browser::text := 'Browser'
   options.browser.text := 'Browser'
   options.browser.type := 'menu'
   options.browser.relief := 'flat'
#
   options.browser.menu := [=];
   options.browser.menu.netscape := [=];
   options.browser.menu.netscape.text := 'Netscape';
   options.browser.menu.netscape.type := 'radio';
   options.browser.menu.netscape.relief := 'flat';
   options.browser.menu.netscape.action := function(){ global helpsystem;
                                       helpsystem::browser := 'netscape';}
#
   options.browser.menu.mosaic := [=];
   options.browser.menu.mosaic.text := 'Mosaic';
   options.browser.menu.mosaic.type := 'radio';
   options.browser.menu.mosaic.relief := 'flat';
   options.browser.menu.mosaic.action := function(){global helpsystem;
                                       helpsystem::browser := 'mosaic';}
#
   options.browser.menu.other := [=];
   options.browser.menu.other.text := 'Other';
   options.browser.menu.other.type := 'radio';
   options.browser.menu.other.relief := 'flat';
   options.browser.menu.other.action := function(){ global helpsystem;
```

```
                                          helpsystem::browser := 'other';}
   return ref options;
}

   # Returns a reference to the default menus, file and options

public.defaultmenus := function()
{  wider self;
   menus := [=];
   menus.file := self.default_filemenu();
   menus.options := self.default_options();
   return ref menus;
}

   # Returns a reference to the default help menu

public.defaulthelp := function()
{
   hmenu := [=];
   hmenu::text := 'Help';
   hmenu.about := [=];
   hmenu.about.text := 'About';
   hmenu.about.relief := 'flat';
   hmenu.about.action := about;
#
   hmenu.refman := [=];
   hmenu.refman.text := 'Reference Manual';
   hmenu.refman.relief := 'flat';
   hmenu.refman.action := function()
                         { help('Refman:'); }
#
   hmenu.gstart := [=];
   hmenu.gstart.text := 'Getting Started';
   hmenu.gstart.relief := 'flat';
   hmenu.gstart.action := function()
                          { help('gettingstarted:'); }

   return ref hmenu;
}

   # Returns a reference to the default actions

public.defaultactions := function()
{  wider public;
   action := [=];
   action.apply := [=];
```

```
   action.apply.text := 'Apply';
   action.apply.action := function(){print 'Apply Pressed'};
   action.reset := [=];
   action.reset.text := 'Reset';
   action.reset.action := function(){print 'Reset Pressed'};
   action.cancel := [=];
   action.cancel.text := 'Cancel';
   action.cancel.action := public.dismiss;
   return ref action;
}
```

Here are a couple of simple examples that illustrate the use of guiframework.
First we construct a simple message window with a dismiss button.

```
# Info dialog

infowindow := function( some_text='Your slogan here.',
                        title='AIPS++ Infomation Window')
{
       #  Create the default action button, Dismiss

    action := [=];
    action.dismiss.text := 'Dismiss';

       #  Create the guiframework, title sets the title of the frame,
       #   the first F, tells the constructor not to use the default menus
       #   the second F, tells the constructor not to use the default help menu
       #   action, says create the action buttons (there's only one)

    a := guiframework(title, F, F, action);

       # We need set the dismiss handler after we create the tool otherwise
       # it wouldn't know how to dismiss itself.

    a.addactionhandler('dismiss', a.dismiss);

       # Now get the frame that will be the parent of all our application
       # gui components

    wf := a.getworkframe();

       # Attach a message to the work frame

    m := message(wf, some_text, width=600);
```

```
      # Return a reference to the info window.
   return ref a;
}
```

Here's a more sophisticated example. It implements a choices window.

```
choicewin := function(question, choices, interactive=have_gui(), timeout=150)
{
    if (!is_string(question) || !is_string(choices)) {
        fail '::choice(description,choices) : description and choices must be strings'
    }

    if (!interactive || !have_gui()) {
        return choices[1]
    }

       #  Setup up the buttons, the default action procedure for each button
       #  is to return its value.

    action := [=];
    for(choice in choices){
       action[choice] := [=];
       action[choice].text := choice;
    }
       # Create the guiframework, no menus, but as many actions as buttons

    a := guiframework("AIPS++ Please make a choice", F, F, action);
    wf := a.getworkframe();

       # Display the query in the workframe

    m := message(wf, question, width=600);

       # Now we setup the timing loop and do a blocking wait for a response

    done := F;
    timer := client("timer", 0.2);
    choiceIs := choices[1];

       # Put the amount of time left to choose in the status loop

    a.updatestatus(spaste('Time left to choose: ', timeout/5, ' s'));

       #  Wait for an answer or until we timeout
    while(!done){
         # Check if a handler has returned
       if(a.handle::returned){
```

```
            # Yup set the choice a break the loop
          choiceIs := a.handle::value;
          break;
       }
          # Nope, wait for the timer
       await timer->ready
       if ($name == "ready") {
  timeout -:= 1;
  done := timeout <= 0;
          timestring := '     ';
          if(!(timeout%5)){
                # Update the status line with the timeleft, count down seconds
             timestring := spaste('Time left to chose: ', timeout/5, ' s');
     a.updatestatus( timestring);
          }
        }
    }

      # All done, cleanup timer and window and return the choice

    send timer->terminate();
    a.dismiss();
    return choiceIs;
}
```

Constructors

guiframework	Constructs the GUI framework, menus, work area, etc...

Functions

addactionhandler	Sets the handler for a menu or command buttion
addactions	Add buttons to command area.
addhelp	Add help menus to the menubar.
addmenus	Add menus to the menubar.
busy	Turns the icon into a watch or pointer. Disables the guiframework frame.
cleanup	removes agents generated by guiframework
defaultactions	Returns the default actions for command buttons.
defaulthelp	Returns the default help menu.
defaultmenus	Returns the default menus.
dismiss	Dismisses the main window.
getworkframe	Returns a reference to the frame where most GUI components are found.
updatestatus	Updates the status line on the window.
updatetitle	Updates the title line on the frame/window.

guiframework - Constructor

Package **display**
Module **guiutils**
Tool **guiframework**

Constructs the GUI framework, menus, work area, etc...

Synopsis

guiframework(title, menus, helpmenu, actions, guiframenote, parent)

Arguments

title	Window title Allowed:Any string Default: AIPS++Window
menus	menu records Allowed:Glish menus records Default: Default file and options menus
helpmenu	help record Allowed:Glish help menu record Default: Default help menus
actions	Action record Allowed:Glish action record Default: Default action buttons
guiframenote	note function Allowed:any note function Default: note
parent	Parent frame Allowed:Frame to attach guiframework Default: F

Description

The guiframework provides a consistent look and feel GUI for gui applications that use it. With it you can specify standard menus, help and actions.
If the parent frame is `F` guiframe work will generate the top-level frame.
Often we need to change the state of button, grey it, change a button label or the like. Rather than provide lots of guiframework functions to do these things, there is a backdoor using the app record. The app datamember of guiframework contains all the GUI elements defined by a guiframework. The scheme
The record `app.mb.btns` contain all the buttons in the menubar, a button is identified by the menu name and the menu item name. For example the 'file'

menu is identified `app.mb.btns.file`, the open option on file is `app.mb.btns.fileopen`, menu items concatinated the menu name and menu item name to identify themselves.
Action buttons belong to the `app.cmd.b` record. A go action would be `app.cmd.b.go`.

Returns
A standard AIPS++GUI framework window

---

guiframework.busy - Function

Package	**display**
Module	**guiutils**
Tool	**guiframework**

Turns the icon into a watch or pointer. Disables the guiframework frame.

Synopsis
busy(busyFlag)

Arguments

busyFlag	Busy Flag Allowed:T or F, T for busy, F for not busy Default: T

---

guiframework.dismiss - Function

Package	**display**
Module	**guiutils**
Tool	**guiframework**

Dismisses the main window.

Synopsis
dismiss()

Returns

Removes the window from the display

---

guiframework.defaultmenus - Function

Package	**display**
Module	**guiutils**
Tool	**guiframework**

Returns the default menus.

Synopsis
defaultmenus()

Returns
Returns a record of the default menus, file & options

---

guiframework.defaulthelp - Function

Package	**display**
Module	**guiutils**
Tool	**guiframework**

Returns the default help menu.

Synopsis
defaulthelp()

Returns
Returns a record of the default help menu.

---

guiframework.defaultactions - Function

Package	**display**
Module	**guiutils**
Tool	**guiframework**

Returns the default actions for command buttons.

Synopsis
defaultactions()

Returns
Returns a record of the default action buttions, Apply, Reset, and Cancel.

---

guiframework.addmenus - Function

Package	**display**
Module	**guiutils**
Tool	**guiframework**

Add menus to the menubar.

Synopsis
addmenus(menus)

Arguments

menus	Record, containing menus Allowed:Any Glish menu record

Returns
Nothing

---

guiframework.addhelp - Function

Package	**display**
Module	**guiutils**
Tool	**guiframework**

Add help menus to the menubar.

Synopsis
addhelp(helpmenu)

Arguments

helpmenu	Help menu record Allowed:Glish menu record

Returns
Nothing

---

guiframework.addactions - Function

Package	**display**
Module	**guiutils**
Tool	**guiframework**

Add buttons to command area.

Synopsis
addactions(actions)

Arguments

actions	Action buttons Allowed:Glish action record

Returns
Nothing

---

guiframework.getworkframe - Function

Package	**display**
Module	**guiutils**
Tool	**guiframework**

Returns a reference to the frame where most GUI components are found.

Synopsis
getworkframe()

Returns

A reference to frame which you use for the parent of GUI components.

---

guiframework.updatestatus - Function

Package	**display**
Module	**guiutils**
Tool	**guiframework**

Updates the status line on the window.

Synopsis
updatestatus(status)

Arguments

status	Status line Allowed:Any string

Returns
Nothing

---

guiframework.updatetitle - Function

Package	**display**
Module	**guiutils**
Tool	**guiframework**

Updates the title line on the frame/window.

Synopsis
updatetitle(status)

Arguments

status	Status line Allowed:Any string

Returns

Nothing

---

guiframework.addactionhandler - Function

Package	**display**
Module	**guiutils**
Tool	**guiframework**

Sets the handler for a menu or command buttion

Synopsis
addactionhandler(button, action)

Arguments

button	button name Allowed:String Default: Valid button name
action	action functions Allowed:Glish function Default: Valid function

Returns
Nothing

Description
Sometimes it's not possible to prescribe a button/menu action before construction of the framework. Addactionhandler allows you do this after the tool is created. A typical example is a menu exit or an action dismiss button. The following code excerpt illustrates adding a command action.

```
action := [=];
action.dismiss := [=];
action.dismiss.text = 'Dismiss';

a := guiframework('My window', F, F, action);
a.addactionhandler('dismiss', a.dismiss());
```

This code excerpt illustrates adding a menu action. Note that for menus we concatenate the menu button names to produce the proper button name. So, the dismiss option of the file menu has a name of filedismiss. If you have cascading menus, you just keep concatenating names, i.e. options, browser, netscape becomes optionsbrowsernetscape.