Astronomical Data Analysis Software and Systems VI
ASP Conference Series, Vol. 125, 1997
Gareth Hunt and H. E. Payne, eds.

The Portable-CGS4DR Graphical User Interface

P. N. Daly

Joint Astronomy Centre, 660 N. A'ohōkū Place, Hilo, HI 96720, USA

Abstract. This paper describes the elements and style of the Portable-CGS4DR graphical user interface built around some extensions to Tcl and the Tk toolkit.

1. Introduction

CGS4DR (Daly 1995) is a suite of tasks designed to reduce and analyze data, automatically, from the UKIRT long-slit spectrometer. These co-operative tasks use a message system (AMS) and a shared memory noticeboard system (NBS). There is also a command line interface (ICL), although it is cumbersome to use for this application.

Tcl (Ousterhout 1994) is an embeddable scripting language and *Tk* is a graphical user interface toolkit. Both have become accepted in a range of disciplines. They are freely available, run on a variety of architectures, and are extensible. These features make them attractive to the astronomical community.

Since the port of CGS4DR to UNIX, the question of a user interface has been reviewed. Several extensions to Tcl (Terrett 1995a,b) have been introduced that provide access to AMS and NBS. It is these extensions that are the basis for the CGS4DR graphical user interface.

2. Extensions to Tcl

The extensions to Tcl are known collectively as STARTCL. They comprise the message system extensions, the noticeboard system extensions, the graphics window manager widget, and the *awish* and *atclsh* utilities (with the extensions embedded therein) for prototyping interfaces to tasks.

2.1. The Message System Extensions

The message system extensions provide access to inter-task communications via the AMS library. Rather than require this level of knowledge, a suite of procedures has been added that provides higher level access to the extensions. These procedures, used in CGS4DR, are best described by analogy with their ICL counterparts as shown, in brief, in Table 1. Note that the *%V* tag is one of a number of possible returns to the user and that there are also ways to communicate with tasks synchronously.

The tasks do not communicate with the user interface directly but go through a relay script, written in Tcl, called *adamMessageRelay*. The reason

Table 1. Icl and the Tcl Extensions for the Message System.

Action	**Commands**
Load task	ICL> loadw /star/bin/cgs4dr/cred4 (cred4_alias) TCL> adamtask $taskname /star/bin/cgs4dr/cred4
Obey action	ICL> send (cred4_alias) obey status TCL> $taskname obey status “” –inform “puts %V”
Cancel action	ICL> send (cred4_alias) cancel reduce TCL> $taskname cancel reduce “” –inform “puts %V”
Set parameter	ICL> send (cred4_alias) set sky_wt 1.0 TCL> $taskname set sky_wt 1.0 –setresponse “puts %V”
Get parameter	ICL> send (cred4_alias) get verbose (value) TCL> $taskname get verbose –getresponse “puts %V”

for this is that AMS hangs if it expects to receive a message when none is in the queue. This would hang the interface, so the relay script handles the communications, enabling the interface to remain responsive.

2.2. The Noticeboard System Extensions

The noticeboard system stores arrays of bytes in shared memory. Each item within a noticeboard has a datatype and dimension, although it is up to the application to handle this information correctly. The NBS extensions to Tcl are shown, again by analogy with their ICL counterparts, in Table 2. In STARTCL—although not in ICL—it is possible to monitor a NBS item, and couple that to a Tcl *trace variable* command to achieve a variety of effects.

Table 2. Icl and the Tcl Extensions for the Noticeboard System.

Action	**Commands**
Put value	ICL> putnbs ((p4_nb_alias)&'.port_0.autoscale') TRUE TCL> nbs put ${P4NoticeBoard}.port_0.autoscale TRUE
Get value	ICL> getnbs ((p4_nb_alias)&'.port_0.title') (txtvar) TCL> set txtvar [nbs get ${P4NoticeBoard}.port_0.title]
Monitoring	ICL> TCL> nbs monitor ${P4NoticeBoard}.port_0.plot_whole tclvar

3. The Graphical User Interface

Using Tcl/Tk and STARTCL, a graphical user interface has been added to CGS4DR in as little as 12,000 lines of code. Various aspects of the interface are documented elsewhere.[1] In essence, there is a separate window for each of the four tasks (one-to-one mapping), plus the graphics widget. Although it is possible to control more than one task from a single window (one-to-many mapping) this has not been implemented in CGS4DR for reasons of clarity and aesthetics. The default style of the CGS4DR interface is shown in Table 3 although these can be re-defined by the user via the file ${HOME}/cgs4dr_configs/cgs4dr.xopts.

Table 3. The Portable-CGS4DR Graphical User Interface Style.

Class	**Item**	**Default Style**
Backgrounds	Generic	Wheat
	Textpane	Snow
	Scrollbar	Lightyellow (inactive)
		Palegreen (active)
	Radiobutton	Wheat (inactive)
		Palegreen (active)
	Checkbutton	Wheat (inactive)
		Grey (active)
	Button	Pink (inactive)
		Palegreen (active)
	Menus	Wheat (inactive)
		Palegreen (active)
Functionality	Keyboard traversal	Enabled (pull-down menus)
	OK button	Do something
	Cancel button	Abort something
	Dismiss button	Abort help
Cursors	Green arrow	Normal
	Orange pirate	Dialogue box is open
	Red watch	Task is busy
	Yellow pencil	Help box is open
Mouse Buttons	MB1:	
	Single click	Do something (generic)
	Double click	Do something (listbox)
	MB2:	
	Single click	Set dynamic default
	Double click	Clears entry widget
	MB3:	
	Single click	Invokes help from WWW page

[1] http://www.jach.hawaii.edu/ukirt_sw/cgs4/cgs4dr/

4. Special Bindings

The control task is responsible for starting, pausing, and stopping the software in response to a user request. The special bindings here are that the START button is replaced by a STOP button when the software is started (and vice-versa). These buttons and the associated PAUSE button provide a number of toggles in the software. So that the user understands the present state, a status label is updated regularly. This label displays either STOPPED on a red background, RUNNING on a green background, or PAUSED on an amber background, utilising the graphical nature of the interface.

The main action for the plotting task is to plot a dataset specified in the DATA entry widget. This widget has been bound to the PLOT button using a carriage return to avoid unnecessary mouse movements. Available plotting surfaces are represented using bitmaps rather than digits. The graphics window manager widget comes supplied with a variable colour palette, cross hair and a PRINT button offering several hardcopy output formats.

The queue manager task is a generic task for storing any time-stamped information and can be used for other jobs not related to CGS4DR. This means, however, that entering a range of reduction commands into the queue invokes a number of OBEY actions in the task (with different strings to write each time). It does this by tumbling through a pair of Tcl scripts while incrementing a counter. The problem is that if the user enters a large number of observations in error, the interface does not return until the sequence is complete which may take some time. To avoid this problem—a pseudo-hang—an INTERRUPT button has been added to grab the widget immediately, thus stopping the sequence of events.

Mouse button 3 is bound to invoke help text. The text invoked is extracted from a WWW page, dynamically, thus ensuring consistency with the Web document and the need to create only one level of help text.

Acknowledgments. D. L. Terrett (RAL) and B. D. Kelly (ROE) are responsible for the STARTCL extensions to Tcl.

References

Daly, P. N. 1995, in ASP Conf. Ser., Vol. 77, Astronomical Data Analysis Software and Systems IV, ed. R. A. Shaw, H. E. Payne, & J. J. E. Hayes (San Francisco: ASP), 375

Ousterhout, J. K. 1994, Tcl and the Tk Toolkit (Reading, MA: Addison-Wesley)

Terrett, D. L. 1995a, SUN/186, Starlink Project, CLRC, UK

Terrett, D. L. 1995b, in ASP Conf. Ser., Vol. 77, Astronomical Data Analysis Software and Systems IV, ed. R. A. Shaw, H. E. Payne, & J. J. E. Hayes (San Francisco: ASP), 395