C32 Reference Manual:
A Constraint Editor

Dario Giuse

December 1994

Abstract

C32 is an object and constraint editor for Garnet objects. It allows Garnet objects to be viewed and edited using a spreadsheet-like interface. New values can be installed in the slots of an object directly, or constraints can be defined among the objects.

This research was sponsored by NCCOSC under Contract No. N66001-94-C-6037, Arpa Order No. B326. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of NCCOSC or the U.S. Government.

1. Overview of C32

C32 [Myers 91] is an object and constraint editor for Garnet objects. It allows Garnet objects to be viewed and edited using a spreadsheet-like interface. Each object is viewed in a panel, where each row corresponds to a slot in the object. The value of a slot can be changed directly, by editing the value shown in the corresponding row. Constraints can be edited textually in separate formula-editing windows.

C32 can be used as a stand-alone tool to create and edit Garnet objects. It is possible, for example, to edit an existing object by typing its name in the title of a C32 panel, or by pointing and clicking on an object with the mouse. In addition, C32 is integrated with Lapidary, which uses it for several editing tasks that used to be handled specially.

Because it uses the spreadsheet paradigm, C32 is highly structured. Each object is represented as a panel, and all panels are organized horizontally in one long window. A horizontal scroll bar allows different panels to be displayed in the window. Panels that contain more than 12 slots are displayed with a vertical scroll bar, so more slots can be made visible as desired.

C32 keeps the display of each panel up to date. When a slot is modified using C32, the values displayed for other dependent slots are modified accordingly. Even if objects are changed from outside C32 (using the Lisp listener or the mouse, for example), their corresponding panels are always kept up to date.

2. Loading C32

To load C32, you load the file "garnet-c32-loader" and then type

```
    (c32:do-go)
```

Note that if you are using Lapidary, you do not need to load C32 explicitly; Lapidary does it automatically.

The function `(c32:do-go)` creates two windows: the C32 Commands window, which contains the main commands used to control C32, and the spreadsheet window. Initially, the spreadsheet window contains a single, empty panel whose title reads "Object name:". The title of this panel can be edited to the name of an object, which is then displayed in the panel.

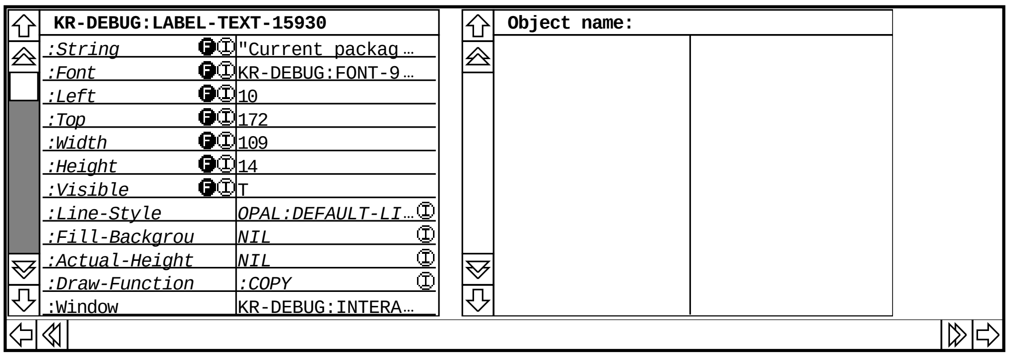
The full syntax for `do-go` is as follows:

`c32:do-go` *&key (startup-objects NIL) (test-p NIL) (start-event-loop-p T)* [*Function*]

If <startup-objects> is specified, it should be a list of Garnet objects. When C32 is started, it creates a panel for each object in the list, plus the empty panel. If <test-p> is specified, a little test window is created with a few objects in it. C32 can then be used to edit the objects in the window. Setting <start-event-loop-p> to NIL cause C32 to start up with the main-event loop not running.

3. The Spreadsheet Window

The spreadsheet window contains a list of panels, each displaying a Garnet object. Figure 1 shows the spreadsheet window with a panel for a `label-text` object and the empty panel. Each panel consists of a vertical scroller (using for displaying more slots), a title, and up to 12 rows. Each row displays the contents of a slot.



KR-DEBUG:LABEL-TEXT-15930		
:String	🅕Ⓘ	"Current packag…
:Font	🅕Ⓘ	KR-DEBUG:FONT-9…
:Left	🅕Ⓘ	10
:Top	🅕Ⓘ	172
:Width	🅕Ⓘ	109
:Height	🅕Ⓘ	14
:Visible	🅕Ⓘ	T
:Line-Style		*OPAL:DEFAULT-LI…* Ⓘ
:Fill-Backgrou		*NIL* Ⓘ
:Actual-Height		*NIL* Ⓘ
:Draw-Function		*:COPY* Ⓘ
:Window		KR-DEBUG:INTERA…

Object name:

Figure 1: C32 Spreadsheet Window with two panels.

The title of each panel shows the name of the object displayed in the panel. The title can be edited by clicking the left mouse button over it, and then using the normal Garnet text editing commands. Type Return to go ahead, and ^G if you do not want to make the change. Entering the name of a different object in a panel's title causes the panel to display the new object. If the object does not exist, C32 will ask you if you wish to actually create a new object. Setting the title of a panel to be empty causes the panel to be removed from the spreadsheet window; the object being displayed is unaffected, however. Setting the title of the empty panel to an object's name causes a new panel to be created for the object.

The left half of each row displays the name of the slot. The names of local slots are shown in a roman face; the names of inherited slots are shown in italics. Slots that contain a formula are indicated by an "F" on a dark circle. If the formula was inherited, this is indicated by an "I" inside a circle, next to the formula symbol. The right half of each row displays the value of the slot. If the value is inherited, an "I" inside a circle is shown at the far right of the row.

At any time, you can have a primary selection and a secondary selection. The primary selection is shown by a dark background, and is used for the majority of C32 operations that require a slot or an object. You may change the primary selection by clicking the left mouse button over a slot name, i.e., in the left side of a row. The secondary selection is shown by a gray outline around a slot, and is used for operations that require two slots. You may change the secondary selection by clicking the middle mouse button over a slot name. Both primary and secondary selections are toggles, and can be eliminated by clicking over them.

Panels allow you to modify objects, as well as displaying them. The value of a slot can be edited by editing its text: first click on the value (in the right half of the row) to get a text cursor, and then use the normal Garnet text editing commands. When you type Return, the slot is set to the new value and the object is modified (type ^G if you do not want to make the change). Note that the package shown in the Commands Window is used when reading the value you type in. Setting the package appropriately ensures that you do not have to type package qualifiers for every function and symbol.

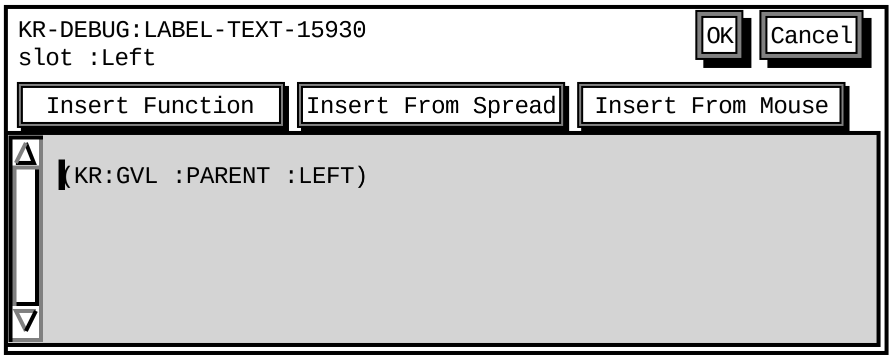
The last row of a panel is always empty. Clicking in its left-hand half allows you to add a slot to the object, or to display a slot that is currently not shown. When you click, you start editing an (initially empty) slot name. If the slot is currently not shown, its current value is displayed. If the slot is not present, it is created. Its initial value is inherited, if possible; otherwise, it is set to NIL. You may then edit the value (in the right-hand side). As a special shortcut, it is also possible to enter a slot name and a value together; just type the slot name, a space, and then the value.

The formula associated with a slot can also be edited using the spreadsheet window. Click on the "F" symbol; this pops up a window that allows the formula to be edited, as explained below. This mechanism also let you create formulas for slots that do not yet contain one: simply click on the place where the "F" symbol would be, and a new formula window will appear. You may then type the text for the new formula. Note that editing the value of a slot that contains a formula is equivalent to doing an `s-value` on the slot with the new value: the old value is temporarily replaced, but the formula is unaffected.

4. Editing Formulas

Formulas can be edited in special editing windows. When you click on the "F" symbol of a slot in the spreadsheet window, a window is created in which you can edit the textual representation of the formula. If you click on the "F" symbol and a window already displays that formula, the window is simply moved to the front. If you click on the "F" symbol of a slot that contains a value, the value is shown as the initial text for the (yet to be created) formula.

Figure 2 shows the C32 formula window for the `:left` slot of the object shown in Figure 1. A formula window contains a header, a vertical scroller, and a text window. The header displays the name of the object and the slot upon which the formula is installed, and contains five buttons. The scroller allows you to examine different portions of long formulas.


KR-DEBUG:LABEL-TEXT-15930
slot :Left

```
(KR:GVL :PARENT :LEFT)
```

Figure 2: The C32 formula window for the :left slot.

When the formula window is created, the text cursor is initially positioned at the top left of the text. The cursor can be moved, and the text can be edited, using the normal Garnet text editing operations. Note that typing the abort character (^G) in a formula window has no effect; click the Cancel button if you really want to abort the current changes to the formula’s text.

The five buttons in the header include the OK and Cancel button, plus three buttons that can be used to reduce typing when the formula is being edited. The "OK" button installs the expression currently displayed in the formula window into the slot of the object, and hides the formula window. If errors are detected (for example, because the syntax in the formula expression is illegal), you will see an error message and the formula window will remain on the screen. The "Cancel" button removes the window without modifying the formula that is currently installed on the slot.

The "Insert Function" button pops up a menu with the names of functions that are commonly used in formulas. In addition to `floor`, `max`, and the like, the menu includes the functions from the `opal:gv-` family. Select a function from the menu by clicking the left mouse button over it; this will show the function’s name in reverse video. Clicking on the "Insert Function" button will now insert the selected function, enclosed in parentheses, at the cursor position in the formula window.

The "Insert From Spread" button inserts a reference to the object and slot that are currently selected in the spreadsheet window into the formula being edited. C32 detects whether the selected object is the same as the one that contains the formula, and if so, it generates a simple reference using `gvl`.

The "Insert From Mouse" button is similar, except that it allows you to select the target object using the control-left mouse button. The button pops up a dialog box through which you may select an object. C32 will attempt to guess a slot; for example, if you click `control-left` on the left part of a string, it will insert the `:left` slot. If C32 cannot guess any slot, it leaves the slot name blank. When the appropriate object (and possibly slot) is shown in the dialog box, clicking the "Apply" button will insert a reference into the formula window. Clicking the "OK" button will do the same, and hide the dialog box as well.

5. The Commands Window

This window contains a menu with C32 commands that apply to the spreadsheet window, and is shown in Figure 3. Many of the buttons in the Commands Window operate on the slot that is currently selected in the spreadsheet window. The window also displays the current package that is used by C32 to interpret Lisp values and expressions (for example, when you type a value or a formula). The package can be changed by editing the string in the box.

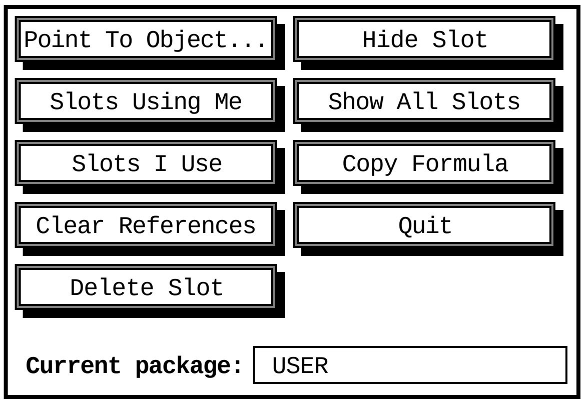


Figure 3: The C32 Commands Window.

The meaning of each group of buttons is explained below.

5.1. Point To Object

This button pops up a dialog box that allows an object to be added to the spreadsheet window by pointing and clicking with the mouse. The dialog box explains how to select objects (by clicking control-left) and how to move from one object to another underneath it. Clicking control-left on any Garnet object inserts the name of the object in the dialog box.

Once the name of the desired object is displayed in the dialog box, you can click the Apply button (which creates a new panel displaying the object), the OK button (which does the same thing and then hides the dialog box), or the Cancel button.

5.2. Showing references to other slots

Three buttons are used to show references, i.e., dependencies among slots. Clicking on the "Slots Using Me" button displays green arrows that indicate what formulas (in objects currently shown in C32) use the selected slot. The arrows originate on the Formula symbol of the dependent slots, and point to the value portion of the selected slot. If a dependent slot belongs to another object, and that object is shown in a panel, the arrow is drawn to the other panel.

Clicking on the "Slots I Use" button of a slot that contains a formula shows red arrows from the formula symbol to all the slots (currently shown in C32) upon which the formula depends. These arrows are heavier than the ones discussed previously, and they point to the slot side, rather than the value side.

Clicking on the "Clear References" button eliminates all currently displayed reference arrows. This operation does not alter any internal value, of course.

5.3. Deleting, hiding, and showing slots

The "Hide Slot" button causes the selected slot to be eliminated from the C32 panel. The value of the slot in the object is unaffected. The "Show All Slots" button causes all slots in the selected object to be displayed in the panel.

The "Delete Slot" is used to delete the currently selected slot from the actual object. Care should be taken, because this is a destructive operation. Trying to delete some of the most important slots prompts for confirmation.

5.4. Copy Formula

The "Copy Formula" button copies the formula installed on the secondary-selected slot to the slot that corresponds to the primary selection. A box pops up to make sure this is what you want to do.

5.5. Quit

This button causes C32 to destroy all its windows and exit. If C32 was started up from Lapidary, however, the windows are simply temporarily hidden, so that C32 can start up faster the next time.

6. C32 Internals

The list of slots to be displayed in a panel is kept in the `:slots-to-show` slot of Garnet objects. In most cases, this slot is inherited. If you are creating new types of objects, you may want to set the `:slots-to-show` slot appropriately in the prototype, so that C32 will display only relevant slots when it shows an instance in a panel.

It is probably a good idea not to try to edit the contents of the `:slots-to-show` slot using C32 itself.

The current version of C32 is not very optimized; redisplaying and scrolling panels, in particular, is rather inefficient. We hope to make its performance better in the next release.

References

[Myers 91] Brad A. Myers.
Graphical Techniques in a Spreadsheet for Specifying User Interfaces.
In *Human Factors in Computing Systems*, pages 243-249. Proceedings SIGCHI'91, New Orleans, LA, April, 1991.

Index

Table of Contents