Gilt Reference Manual: A Simple Interface Builder for Garnet

Brad A. Myers

October 1993

Abstract

Gilt is a simple interface layout tool that helps the user design dialog boxes. It allows the user to place pre-defined Garnet gadgets in a window and then save them to a file. There are two versions: one for Garnet look-and-feel gadgets and one for Motif look-and-feel gadgets.

This research was sponsored by the Avionics Lab, Wright Research and Development Center, Aeronautical Systems Division (AFSC), U. S. Air Force, Wright-Patterson AFB, OH 45433-6543 under Contract F33615-90-C-1465, Arpa Order No. 7597. Additional support for Garnet has been provided by NEC, Apple, Adobe, and General Electric.

1. Gilt

1.1. Introduction

This document is the reference manual for the *Gilt* tool, which is part of the Garnet User Interface Development System [Myers 90]. Gilt stands for the Garnet Interface Layout Tool, and is a simple interface builder for constructing dialog boxes. A dialog box is a collection of *gadgets*, such as menus, scroll bars, sliders, etc. Gilt supplies a window containing many of the built-in Garnet gadgets (see Figure 1), from which the user can select the desired gadgets and place them in the work window. Gilt does *not* allow constraints to be placed on objects or for new gadgets or application-specific objects to be created.

There are two sets of gadgets in Gilt. Each allows you to create dialog boxes with a consistent look-and-feel. The standard Garnet gadgets are shown in Figure 1, and the Motif style gadgets are in Figure 2). Both versions operate the same way. You can toggle between the standard and Motif gadget palettes by selecting "`Load Other Gadgets`" from the main Gilt menubar.

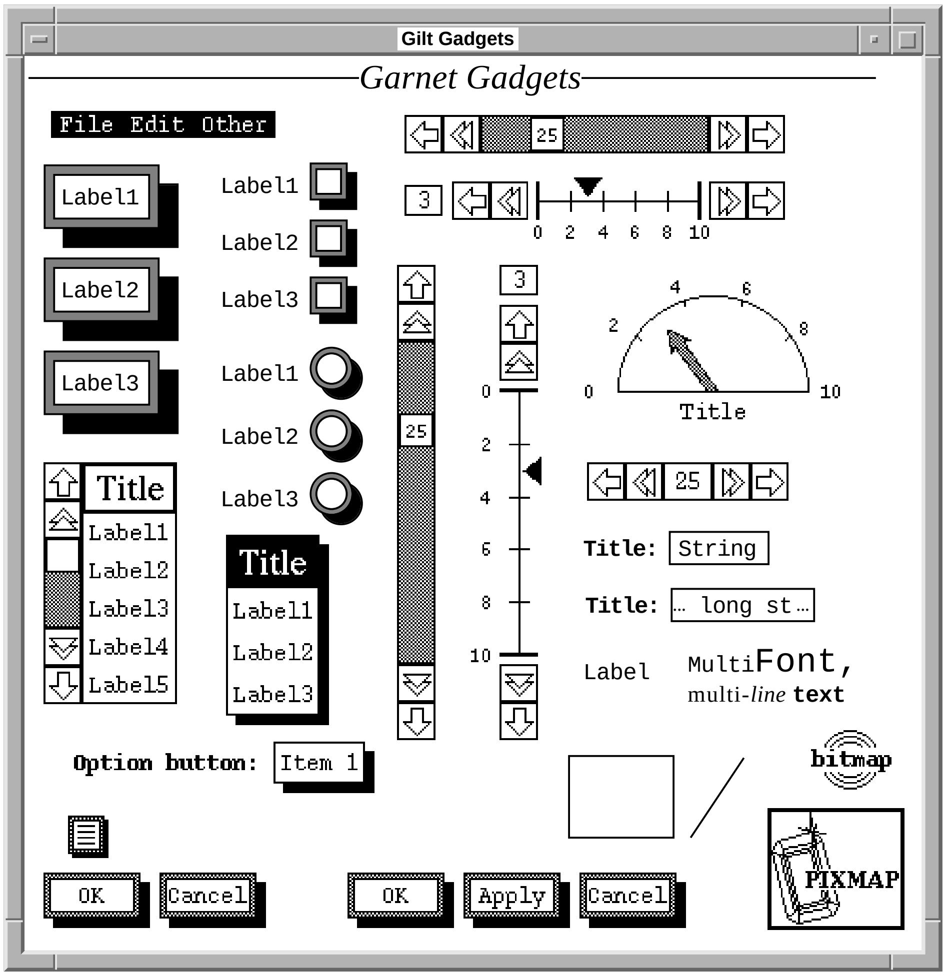

Figure 1: The Gilt gadget window for the Garnet look and feel. All of the gadgets that can be put into the window are shown. The check boxes are selected.

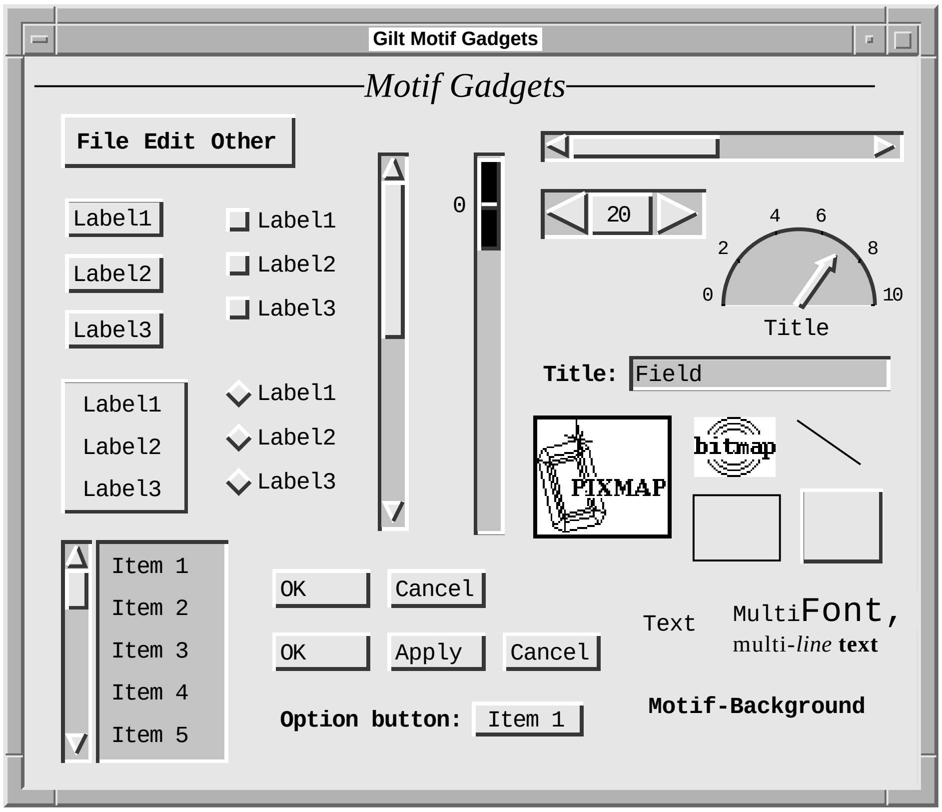

Figure 2: The Gilt gadget window for the Motif look and feel.

There is a more powerful interactive design tool in Garnet called Lapidary [Myers 89]. Lapidary allows new gadgets to be constructed from scratch, and allows application-specific graphics to be created without programming. However, Lapidary does not support the placement of the existing Garnet gadgets.

1.2. Loading Gilt

Gilt is *not* automatically loaded when you load Garnet. After Garnet is loaded, to load Gilt do:

```
(load Garnet-Gilt-Loader)
```

There is only one version of Gilt, but you can specify what set of gadgets should appear in the palette when the windows appear. This is determined by a required parameter to `do-go`. To start Gilt, do:

```
(gilt:do-go :motif)
or
(gilt:do-go :garnet)
```

Gilt can be stopped by selecting "`Quit`" from the menubar, or by executing `(gilt:do-stop)`.

1.3. User Interface

Gilt displays three windows: The gadgets window, the main command window, and the work window. The main command window is shown in Figure 3. Figure 4 shows an example session where the work window contains gadgets with the Garnet look-and-feel. The two types of gadget palette windows are shown in Figures 1 and 2.

For the Garnet look and feel, examples are in Figures 1, 3 and 4. Figure 2 shows the Gadget window for the Motif look and feel.

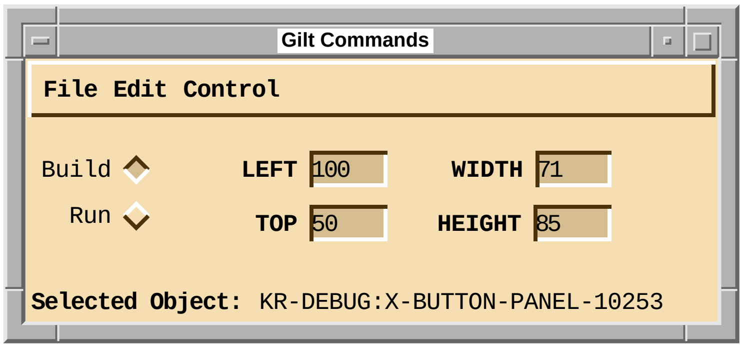

Figure 3: The Gilt Command window for the Garnet look-and-feel (the Motif one is similar). The control groups are (top to bottom, left to right): whether building or running the interface, the coordinates of the selected object, the name of the selected object, and the menu of commands.

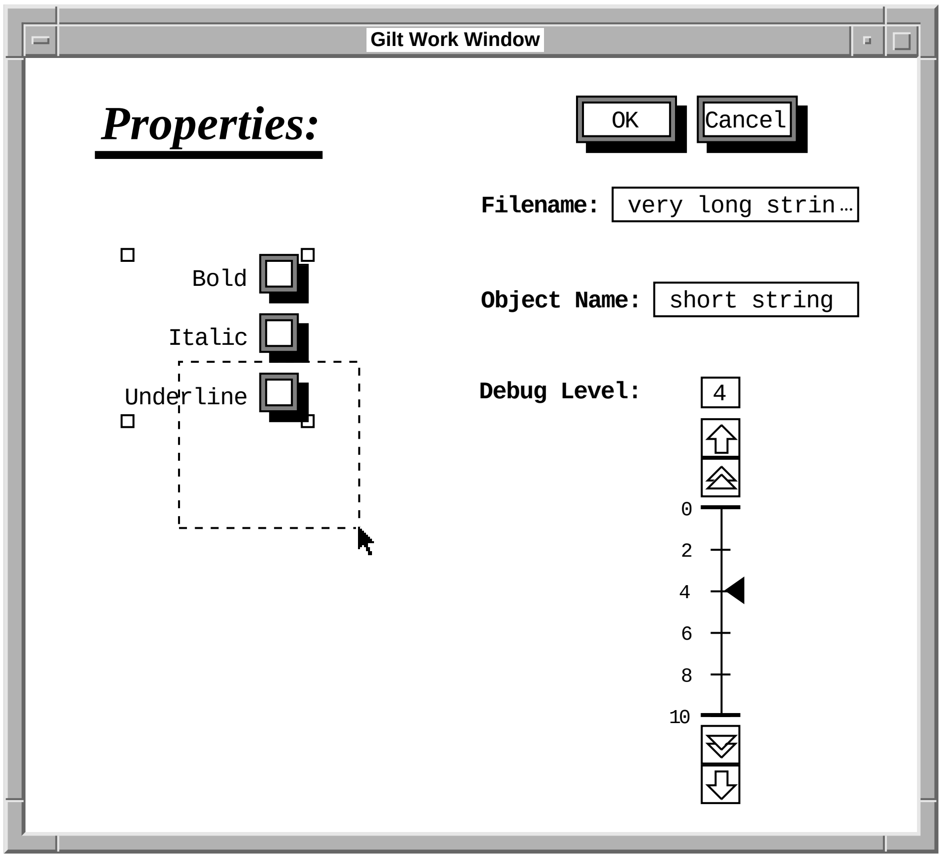

Figure 4: The Gilt Work window showing a sample dialog box being created using the Garnet look and feel.

1.3.1. Gadget Palettes

The single version of Gilt allows you to place Motif or Garnet look-and-feel gadgets into your window (you can mix and match if you want). To switch back and forth, use the "`Load Other Gadgets`" command in the "`File`" menu of the Gilt menubar. You can only see one gadget window at a time. The dialog boxes for Gilt itself use only the Motif look and feel.

1.3.2. Placing Gadgets

When you press with any mouse button on a gadget in the gadgets palette window (see Figures 1 or 2), that gadget becomes selected. Then, when you press with the *right* mouse button in the work window, an instance of that gadget will be created. Some gadgets, such as the scroll bars, have a variable size in one or more dimensions, so for those you need to press the right button down, drag out a region, and release the button.

The gadgets supplied for the *Garnet* look and feel are (from top to bottom, left to right in Figure 1):

- Menubar: a pull-down menu,
- Text-button-panel: for commands,
- Scrolling-menu: when there are many items to choose from,
- Option-button: a popup-menu which changes the label of the button according to the selected item,
- Popup-menu-button: a popup-menu which does not change labels,
- OK-Cancel: A special gadget to be used when you want the standard OK and Cancel behavior (see section 1.8.1),
- X-button-panel: for settings where more than one is allowed,
- Radio-button-panel: for settings where only one is allowed,
- Menu: a menu with an optional title,
- H-scroll-bar: for scrolling horizontally,
- H-slider: for entering a number in a range,
- V-scroll-bar: for scrolling vertically,
- V-slider: for entering a number in a range,
- OK-Apply-Cancel: Similar to OK-Cancel, but supports Apply (like OK, but don’t remove window),
- Gauge: another way to enter a number in a range,
- Trill-device: enter a number either in a range or not,
- Labeled-box: enter any string; box grows if string is bigger,
- Scrolling-labeled-box: enter a string; box has a fixed size and string scrolls if too big,
- Text: for decoration,
- Multifont-text: for decoration,
- Rectangle: for decoration,
- Line: for decoration,
- Bitmap: for decoration,
- Pixmap: for decoration.

The gadgets supplied for the *Motif* look and feel are (from top to bottom, left to right in Figure 2):

- Motif-Menubar: a pull-down menu
- Motif-Text-button-panel: for commands,
- Motif-Menu: a menu with an optional title,
- Motif-Scrolling-Menu: when there are many items to select from,
- Motif-Check-button-panel: for settings where more than one is allowed,
- Motif-Radio-button-panel: for settings where only one is allowed,
- Motif-OK-Cancel: A special gadget to be used when you want the standard OK and Cancel behavior (see section 1.8.1),
- Motif-OK-Apply-Cancel: similar to OK-Cancel, but supports Apply,

- Motif-Option-Buton: a popup-menu whose button's label changes according to the selection
- Motif-V-scroll-bar: for scrolling vertically,
- Motif-V-slider: for entering a number in a range,
- Motif-H-scroll-bar: for scrolling horizontally,
- Motif-trill-device: for selecting from a range of numbers,
- Motif-Gauge: another way to enter a number in a range,
- Motif-Scrolling-labeled-box: enter a string; box has a fixed size and string scrolls if too big,
- Pixmap: for decoration
- Bitmap: for decoration
- Rectangle: for decoration,
- Line: for decoration,
- Motif-Box: A gadget that resembles a raised (or depressed) rectangle, used to achieve a Motif style effect. Set the `:depressed-p` parameter.
- Text: for decoration,
- Multifont-text: for decoration,
- Motif-Background: a special rectangle that helps achieve the Motif effect. It always moves to the back of the window, and can only be selected at the edges.

In addition to the standard gadgets, Gilt supplies a text string, a line, a rectangle and a bitmap. These are intended to be used as decorations and global labels in your dialog boxes. They have no interactive behavior.

The Motif version also provides a background rectangle. This is a special rectangle which you should put behind your objects to make the window be the correct color. Note: to select the motif-background rectangle, press at the edge of the window (the edge of the background rectangle). You might want to select the rectangle to delete it or change its color (using the properties menu).

1.3.3. Selecting and Editing Gadgets

When you press with the left mouse button on a gadget in the work window, it will become selected, and will show four or twelve selection handles. The objects that can change size (such as rectangles and scroll bars) display black and white selection handles, and the objects that cannot change size (such as buttons) only show white selection handles.[1] If you press on a *white* handle and drag, you can change the object's position. If you press on a *black* handle, you can change it's size (see Figure 5).

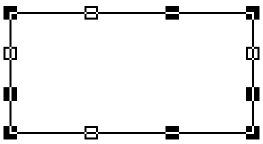

Figure 5: Pressing on a black selection handle causes the object to grow, and pressing on a white one causes it to move.

If you press over an object with either the *middle* mouse button or hold down the keyboard shift key while hitting the left button, then that object is added to the selection set (so you can get multiple items selected). If you press with middle or shift-left over an item that is *already* selected, then just that item

[1]You can indirectly change the size of buttons by setting offsets and sizes in the property sheet, however.

becomes de-selected. If you press with the left button in the background (over no objects), all objects are deselected. While multiple objects are selected, you can move them all as a group by pressing on the selection handle of any of them. Since Gilt uses the multi-selection gadget, it supports selecting all objects in a region (hold down the left button and sweep out a region), and growing multiple selected objects (if they are growable, then press on a black handle at the outside of the set of objects).

To explicitly set the size or position of the selected object (when only one is selected), you can use the number fields in the Command Window (see Figure 3). Simply press with the left button in one of these fields and type a new number. When you hit `return`, the object will be updated. These fields are a handy way to get objects to be evenly lined up (but also see the "`Align`" command).

1.4. Editing Strings

Editing the strings of most gadgets is straightforward: select the gadget (to get the selection handles around it) and then click in a string to get the string cursor, and then type the new string, and hit `return` when done. If you make the string empty (e.g., by typing `control-u`), and hit `return`, that button of the gadget will be removed. If you edit the last item of the gadget and hit `control-n` instead of `return`, then a new item will be added to the gadget. The strings can also be edited by editing the `:items` property in the property sheet that appears from the `Properties` command.

To edit string labels, simply click to select them, and then click again with the left button to begin editing. The fonts of multifont strings can be edited using the keyboard commands described in the "Multifont" section of the Opal Manual.

To edit the strings in a pop-up menu, like a menubar or an option button, click once with the left button to select the gadget, and then click again to pop-up the submenu. You can now click in the submenu to edit any of the items. Use control-n in the last item to add new items or control-u and return to remove items. To edit the top-level labels of a menubar, you need to click the left button three times: once to select the gadget, once to bring up the submenu, and a third time to begin editing. Click outside to make the popped-up menu disappear.

The editing operations supported for regular text (and labels) are:

`^h, delete, backspace`: delete previous character.
`^w, ^backspace, ^delete`: delete previous word.
`^d`: delete next character.
`^u`: delete entire string.
`^b, left-arrow`: go back one character.
`^f, right-arrow`: go forward one character.
`^a`: go to beginning of the current line.
`^e`: go to end of the current line.
`^y`: insert the contents of the X cut buffer into the string at the current point.
`^c`: copy the current string to the X cut buffer.
`enter, return, ^j, ^J`: Finished.
`^n`: Finished, but add a new item (if a list).
`line-feed`: Start a new line (if editing a multi-line text).
`^g`: Abort the edits and return the string to the way it was before editing started.

If the item is a member of a list, such as a menu item or a radio button, then if the string is empty, that item will be removed. If the string is terminated by a `^n` (control-n) instead of by a return, and if this is the last item, then a new item will be added. The items can also be changed in the properties dialog box for the gadget (see below).

Some strings cannot be edited directly, however. This includes the labels of sliders and gauges, and the indicators in scroll bars. To change these values, you have to use the property sheets. Also, for gadgets that have strings as their *values*, such as the text input field and scrolling-text input field, you can only set the value strings by going into Run mode. Note, however, that the values are not saved with the gadget (see section 1.8).

To change the bitmap picture of a bitmap object, specify the name of the new bitmap using the "`Properties...`" command.

1.5. Commands

There are many commands in Gilt, and the command menu is a menubar at the top of the main window. The menubar implementation allows you to give commands using keyboard shortcuts when the mouse is in the main Work Window. The particular shortcuts are listed on the sub-menus of the main menubar.

The commands are:

`Cut` — remove the selected item(s) but save them in the clipboard so they can later be pasted.

`Copy` — copy the selected item(s) to the clipboard so they can later be pasted.

`Paste` — place a copy of the items in the clipboard onto the window.

`Duplicate` — place a duplicate of the selected items onto the window. (See section 1.5.2.)

`Delete` — delete the selected objects and don't put them into clipboard. This operation can be undone with the `Undo Last Delete` command. (See section 1.5.4.)

`Delete All` — delete all the objects in the window. This operation can be undone with the `Undo Last Delete` command. (See section 1.5.4.)

`Undo Last Delete` — undoes the last delete. All the deletes are saved, so this command can be executed multiple times to bring back objects deleted earlier. (See section 1.5.4.)

`Select All` — select all the objects in the window (including the background object).

`To Top` — make the selected objects not be covered by any other objects. (See section 1.5.1.)

`To Bottom` — make the selected objects be covered by all other objects. (See section 1.5.1.)

`Properties...` — bring up the properties window. (See section 1.5.5).

`Align` — bring up the dialog box to allow aligning of the selected objects with respect to the first of the objects selected. (See section 1.5.3.)

Many of these commands are now implemented with the functions in the `Standard-Edit` mechanism, described in the Gadgets Manual.

1.5.1. To-Top and To-Bottom

The selected object or objects can be made so they are not covered by any objects using the "`To Top`" command in the Gilt Command Window. The objects can be made to be covered by all other objects by selecting the "`To Bottom`" command.

1.5.2. Copying Objects

The "`Duplicate`" command in the Command Window causes the selected object or objects to be duplicated. The new object or objects will have all the same properties as the original, but the original and new objects can be subsequently edited independently without affecting the other object (the new object is a copy, not an *instance* of the original). The copy is placed at a fixed offset below and to the right of the original, and is selected, so it can subsequently be moved.

1.5.3. Aligning Objects

The Align function allows you to neatly line up a set of objects, and to adjust their sizes to be the same. Figure 6 shows the dialog box that appears when the "`Align...`" command is selected. Align adjusts the present positions of objects only; it does not set up constraints. Therefore, you can freely move objects after aligning them.

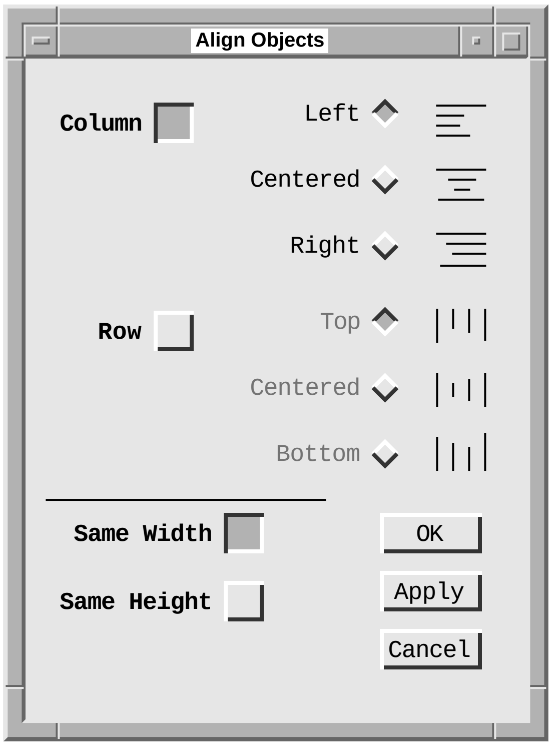

Figure 6: The Align dialog box, after the user has specified that the selected objects should be aligned and centered in a column and be adjusted to be the same width.

To use Align, you first select two or more objects in the workspace window (remember, to select more than one object, press on the objects with the middle mouse button or hold down the shift key while hitting the left button). The *first* object you select is the reference object, and the other objects will be adjusted with respect to that first object. For example, if you want to make objects be the same width, then the width will be that of the first selected object. You should not change the selection while the Align dialog box is visible.

Aligning in a column or row also adjusts the spacing between objects to be all the same. The spacing used between the objects is the average space between the objects before the command is given.

If a line is selected, then it is made to be exactly horizontal if "Column" is specified, or vertical if "Row" is selected. The size of lines can also be adjusted using the width and height buttons. If both "Same Width" and "Same Height" are selected for a line, then an error message is given.

If the "Same Width" and/or "Same Height" buttons are pressed, and one of the selected objects other than the first cannot change size, then an error message is presented. All other selected objects are still adjusted, however.

1.5.4. Deleting Objects

Choosing the "`Delete Selected`" command in the Gilt Command Window will remove the selected object or objects from the work window. Selecting the "`Undo Last Delete`" command will bring the object back. Selecting "`Delete All`" removes all the objects from the work window. "`Undo Last Delete`" will bring all of the objects back. All of the deleted objects are kept in a queue, so the undo command can be executed repeatedly. Note that this is not a general Undo; only undoing of deletes is supported.

1.5.5. Properties

Each type of gadget has a number of properties. First select the object in the workspace window, and then select the "`Properties...`" command (in the Gilt Command Window). The window for the properties will appear below the selected object, but then can be moved. You should not change the selection while the properties dialog box is visible.

If the selected object is a rectangle, line or string, then special dialog boxes are available so you can change the color, filling-style, font, etc. These were created using Gilt.

The general property sheet lists all of the properties that you can change, and will look something like Figure 7. You can press in the value (right) side of any entry and then type a new value (using the same editing commands as in section 1.4). You can move from field to field using the tab key (after pressing with the left mouse button in a field to start with). When finished setting values, hit the "OK" button to cause the values to be used and the property sheet to disappear, or hit the "Apply" button to see the results and leave the property sheet visible. If you hit "Cancel", the changes will not affect the object, and the property sheet will go away.

You can select multiple items and bring up a property sheet on all of them. The property sheet will show the union of all properties of all objects. If multiple objects have the same property name, then the value of the property for the first object selected is shown.

When you edit the value of a property and then hit return (or when you hit OK for properties that pop up dialog boxes), the property sheet will immediately set that property into all objects for which the property is defined. Thus, you can change the `:foreground-color` of all the objects by executing `Select All`, bringing up the `Properties...`, and then editing the foreground-color property. If you start to edit a property but change your mind, hit `Control-G` if text editing or `Cancel` in a dialog box. The `Done` button hides the property sheet.

The left, top, width and height number boxes displayed in the main Gilt window will now also work on multiple objects. When multiple objects are selected, they show the values for the bounding box of all the objects, and when you edit one and hit RETURN, that value is applied to all objects for which it is settable.

For a complete explanation of what the fields of each gadget do, see the Gadgets Manual.

Some of the fields of these property sheets are edited in a special way. The `DIRECTION` field must be either `:VERTICAL` or `:HORIZONTAL`, so the field shows these names, and you can press with the left button to pick the desired value. Fields that represent fonts show a special icon, and if you click on it, the special font dialog box will appear. However, the font is not changed in the object until the "OK" or "Apply" buttons are hit on *both* the font dialog box and the main property sheet.

The field named `KNOWN-AS` should be set for all gadgets that programs will want to know the values of, and will be the name of the slot that holds the object (so it should be a keyword, e.g., `:myvalue`). The `SELECT-FUNCTION` slot can contain a function to be called at run time when the gadget is used. Note that you might want to specify the package name on the front of the function name. However, if you are going to have OK-Cancel or OK-Apply-Cancel in the dialog box, you probably do not want to supply

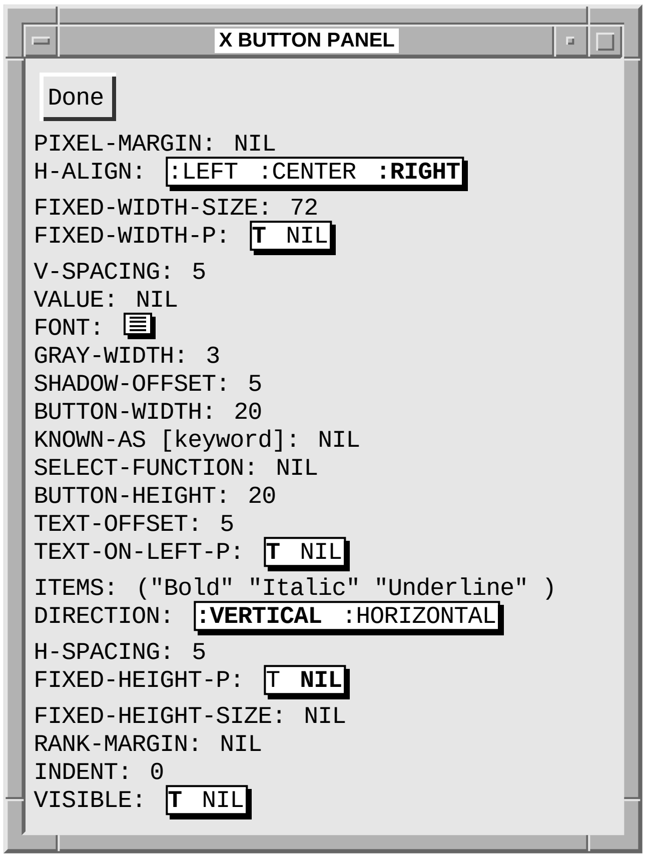

Figure 7: The property sheet that appears for a particular X-Button-Panel.

selection functions, since selection functions are called when the gadget is used, not when OK is hit (see section 1.9).

If the property sheet thinks any value is *illegal*, the value will be displayed in italics after a return or tab is hit, and Gilt will beep. You can edit the value, or just leave it if the value will become defined later (e.g., if the package is not yet defined).

Unfortunately, however, the error checking of the values typed into the property sheets is not perfect, so be careful to check all the values before hitting OK or Apply. If a bad value is set into the gadget, Gilt will crash. You can usually recover from this by setting the field back to a legal value in the Lisp window. For example, if `:gray-width` got a bad value, you might type:

```
(kr:s-value user::*gilt-obj* :gray-width 3)
(opal:update-all)
(inter:main-event-loop)
```

1.5.6. Saving to a file

When the "`Save...`" command is selected from the Command window, Gilt pops up the dialog box shown in Figure 8.

The only field you need to fill in is the "`Filename`" field, which tells the name of the file that should be written. Simply press with the left button in the field and begin typing. This is a scrollable field, so if the name gets too long, the text will scroll left and right. You might also want to use the window manager’s

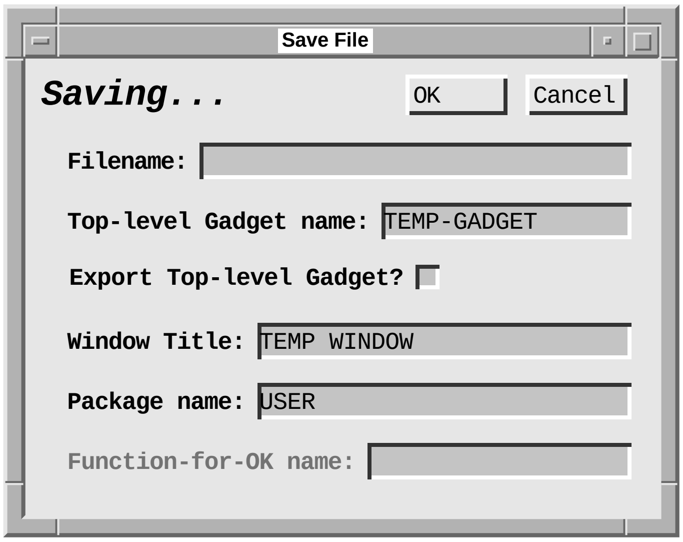

Figure 8: The dialog box that appears when the Save command is chosen.

cut buffer (^Y) if you can select the string for the file in a different window. Pressing with the mouse button again will move the cursor, so you need to hit `return` or `^G` to stop editing the text field.

All the objects in the work window will be collected together in a single Garnet ‘‘aggregadget’’ when written to the file. The "`Top-level Gadget name`" field allows you to give this gadget a name. This is usually important if you want to use the gadget in some interface, so you can have a name for it. If you press the "`Export Top-level Gadget`" button, then an export line will be added to the output file.

As described below in section 1.9, there is a simple function for displaying the created gadget in a window. If you want this window to have a special title, you can fill this into the "`Window Title`" field. The current position and size of the workspace window is used to determine the default size and position of the dialog box window when it is popped up, so you should change the workspace window’s size and position (using the standard window manager mechanisms) before hitting OK in the Save dialog box.

If you want the gadget to be defined in a Lisp package other than `USER`, then you can fill this into the "`Package name`" field.

Finally, if you have included the special OK-Cancel gadget in your workspace window, then the "`Function-for-OK name`" field will be available. Type here the name of the function you want to have called when the OK button is hit. The parameters to this function are described in section 1.9.

After filling in all the fields, hit "OK" to actually save the file, or "Cancel" to abort and not do the save.

If you have already read or saved a file, then the values in the Save dialog box will be based on the previous values. Otherwise, the system defaults will be shown.

Note: There is no protection or confirmation required before overwriting an existing file.

1.5.7. Reading from a file

You can read files back into Gilt using the "`Read...`" command. This displays the dialog box shown in Figure 9. Press with the left mouse button in the "`Filename`" field and type the name of the file to be read, then hit return.

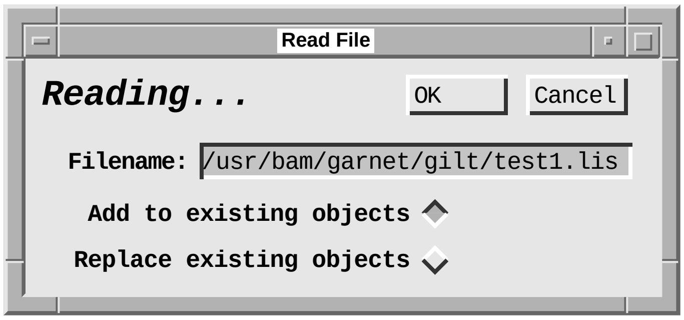

Figure 9: The dialog box that appears when the Read command is hit.

If there are objects already in the workspace window, then you have the option of adding the objects in the file to the ones already in the work window using the "`Add to existing objects`" option, or else you can have the contents of the workspace window deleted first using the "`Replace existing objects`" option. If you use the "`Replace`" option, then the window size is adjusted to the size specified when the file was written. Also, reading a file using the replace option puts the previous contents of the workspace window in the delete stack so that they can be retrieved using the "`Undo Last Delete`" command.

Output produced from the GarnetDraw utility program can be read into Gilt, which would allow more elaborate decorations to be added to a dialog box. But in general, only files written with Gilt can be read with Gilt.

1.5.8. Value and Enable Control

A sophisticated module for modifying the values of gadgets in the Gilt work-window has been added, along with a corresponding module to modify when a gadget should be active (or grayed-out). These are called the `Value Control` and `Enable Control` modules, and can be invoked from the "Control" submenu in the Gilt menubar.

These modules implement the ideas discussed in [Myers 91]. The paper includes examples of how to use this feature, but a full set of documentation is still pending. If there is sufficient demand for documentation of this module, we will supply an addendum to this manual (direct requests to `garnet@cs.cmu.edu`).

1.6. Run Mode

To try out the interface, just click on the button in the command window labeled "Run". This will grey out most of the commands, and allow the gadgets in the work window to execute as they will for the end user (except that application functions will not be called). To leave run mode, simply press on the "Build" button.

1.7. Hacking Objects

Gilt does not provide all options for all objects and gadgets. If you want to change other properties of objects that are not available from the property sheets, you could hit the HELP key while the mouse is positioned over the object to bring up the Inspector (see the Debugging manual, starting on page 469 for details).

You can also access the selected object directly from Lisp. If one object is selected, its name is printed in the command window. Also the variable `user::*gilt-obj*` is set with the single selected object. If multiple objects are selected, then `user::*gilt-obj*` is set with the list of objects selected. You can go into the Lisp listener window, and type Garnet commands to affect the selected object (e.g., `s-value` some slots), and call `(opal:update-all)`. This technique can also be used to add extra slots to objects. The changes you make will be saved with the object when it is written.

1.8. Using Gilt-Created Dialog Boxes

There are various ways to use Gilt-created collections of gadgets in an application.

The file that Gilt creates is a normal Lisp text file that creates the appropriate Garnet objects when loaded. The file should be compiled along with your other application files, in order to provide better performance.

1.8.1. Pop-up dialog box

Probably the easiest way to use a set of gadgets is as a pop-up dialog box. The application should be sure to *load* the file that Gilt created before calling the functions below.

When Gilt writes out the gadgets, it does *not* save the values as the initial defaults. Therefore, if you want to have default values for any gadgets, you need to set them from your program. This should be done *before* the window is displayed using the function:

`gilt:Set-Initial-Value` *top-gadget gadget-name value* [*Function*]

The `top-gadget` is the top-level gadget name specified in the "`Top-level Gadget name`" field of the Save dialog box. The `gadget-name` is the name of the particular gadget to be initialized. This name will be a keyword, and will have been specified as the `KNOWN-AS` property of the gadget using the gadget's property sheet (which appears when you hit the "`Properties...`" command). The `value` is the value to be used as the default, in the appropriate format for that gadget.

Next, the Gilt function `show-in-window` can be used to display the dialog box in a window:

`gilt:Show-In-Window` *top-gadget* `&optional` *x y modal-p* [*Function*]

The *top-gadget* is the gadget name used in the Save dialog box. The size of the window is determined by the size of the workspace window when the file was written. The position of the window will either be the position when written, or it can be specified as the *x* and *y* parameters, which are relative to the screen's upper-left corner. When the *modal-p* parameter is T, then interaction in all other Garnet windows will be suspended until the window goes away (e.g., when the user clicks the "OK" button). If you want the window relative to a position in another window, the function `opal:convert-coordinates` is useful.

The function `show-in-window-and-wait` performs the same function as `show-in-window`, but it waits for the user to click on an OK or Cancel button before returning (`show-in-window` returns immediately after bringing up the window).

`gilt:Show-In-Window-And-Wait` *top-gadget* `&optional` *x y modal-p* [*Function*]

When the user clicks on the OK button, this function will return the values of all the gadgets in the dialog box in the form of `gilt:gadgets-values`, which is:

```
((:FILENAME "/usr/bam/garnet/t1.lisp") (:VAL 49) (:BUTTON "Start"))
```

where the keywords are the names (`:known-as` slot) of the gadgets. If the user hits Cancel, then `show-in-window-and-wait` returns NIL. Apply does not cause the dialog box to go away, so you might want to supply an OK-Function for the dialog box.

If selection functions were specified in the gadget's select-function slot using the "`Properties...`" command, then these functions are called immediately when the gadgets are used.

If the dialog box has an OK-Cancel or OK-Apply-Cancel gadget in it, then the function specified in the "`Function-For-OK name`" field of the Save dialog box will be called when the user hits the OK or Apply buttons. This function is parameterized as:

```
(lambda (top-gadget values)
```

The `top-gadget` is the same as above. The `values` parameter will be a list containing pairs of all the gadget names of gadgets which have names, and the value of that gadget. Again, the names are the keywords supplied to the `KNOWN-AS` property. For example, `values` might contain:

```
((:FILENAME "/usr/bam/garnet/test1") (:REINITIALIZE NIL))
```

The function

`gilt:Value-Of` *gadget-name values* [*Function*]

can be used to return the value of the specific gadget named `gadget-name` from the values list `values`. For example, if `v` is the above list, then `(gilt:value-of :filename v)` would return `"/usr/bam/garnet/test1"`.

After the Function-For-OK is called, the dialog box window is made invisible if OK was hit, and left in place if Apply was hit. If Cancel was hit, then the window is simply made invisible. If `show-in-window` is called again on the same dialog box, the old window is reused, saving time and space.

To destroy the window associated with a gadget, use the function:

`gilt:Destroy-Gadget-Window` *top-gadget* [*Function*]

This does *not* destroy the `top-gadget` itself. Note that destroying the top-gadget while the window is displayed will not destroy the window. However, destroying the window explicitly (using `opal:destroy`) *will* destroy both the window and the gadget.

1.9. Using Gilt-Created Objects in Windows

If you want to use the gilt-created gadgets inside of an application window, you only need to create an instance of the top-gadget, which is the top-level gadget name specified in the "`Top-level Gadget name`" field of the Save dialog box. The instance will have the same position in the application window as it had in the Gilt workspace window. If you use the standard Gilt OK-Cancel gadget, it will make the application window be invisible when the OK or Cancel buttons are hit. If you do not want this behavior, then you need to create your own OK-Cancel buttons.

The `set-initial-value` described above can still be used for gilt gadgets in application windows. In addition, the function

`gilt:Gadget-Values` *top-gadget* [*Function*]

can be used to return the values list of all gadgets with names. The return value is in the form of the `values` parameter passed to the Ok-Function.

1.10. Hacking Gilt-Created Files

Since the file that Gilt creates is a normal text file, it is possible to edit it with a normal text editor. Some care must be taken when doing this, however, if you want Gilt to be able to read the file back in for further editing. (If you do not care about reading the files back in, then you can edit the file however you like.)

The simplest changes are to edit the values of slots of the objects. These edits will be preserved when the file is read in and written back out. Be sure not to change the value of the `:gilt-ref` slot.

When Gilt saves objects to a file, it sets the `:constant` slot of all the gadgets to T. If you expect to ever change any properties of widgets in the dialog boxes when they are being used by an application, then you should hand-edit the Gilt-generated file to change the constant value (typically by `:excepting` the slots you plan to change dynamically). (Gilt reads in the file using `with-constants-disabled`, so the defined constant slots will not bother Gilt.)

If you want to create new objects, then these can be put into the top level aggregadget definition. You should follow the convention of having a `:box` (or `:points`) slot and putting the standard constraints into the `:left` and `:top` fields (or `:x1, :y1, :x2` and `:y2`). For example, to add a circle, the following code might be added into the top-level aggregadget's `:parts` list:

```
(:mycircle ,opal:circle
  (:box (50 67 30 30))
  (:left ,(o-formula (first (kr:gvl :box))))
  (:top ,(o-formula (second (kr:gvl :box))))
  (:width 30)
  (:height 30))
```

If you do not supply a `:gilt-ref` field, Gilt will allow the user to move the object around, but not change its size or any other properties. For some objects, it might work to specify the `:gilt-ref` slot as `"TYPE-RECTANGLE"`, `"TYPE-LINE"` or `"TYPE-TEXT"`.

If you add extra functions or comments to the file, they will *not* be preserved if the file is written out and read in. Similarly, interactors added to the top level gadget will not be preserved.

References

[Myers 89] Brad A. Myers, Brad Vander Zanden, and Roger B. Dannenberg.
Creating Graphical Interactive Application Objects by Demonstration.
In *ACM SIGGRAPH Symposium on User Interface Software and Technology*, pages 95-104. Proceedings UIST'89, Williamsburg, VA, November, 1989.

[Myers 90] Brad A. Myers, Dario A. Giuse, Roger B. Dannenberg, Brad Vander Zanden, David S. Kosbie, Edward Pervin, Andrew Mickish, and Philippe Marchal.
Garnet: Comprehensive Support for Graphical, Highly-Interactive User Interfaces.
IEEE Computer 23(11):71-85, November, 1990.

[Myers 91] Brad A. Myers.
Separating Application Code from Toolkits: Eliminating the Spaghetti of Call-Backs.
In *ACM SIGGRAPH Symposium on User Interface Software and Technology*, pages 211-220. Proceedings UIST'91, Hilton Head, SC, November, 1991.

Index

Table of Contents