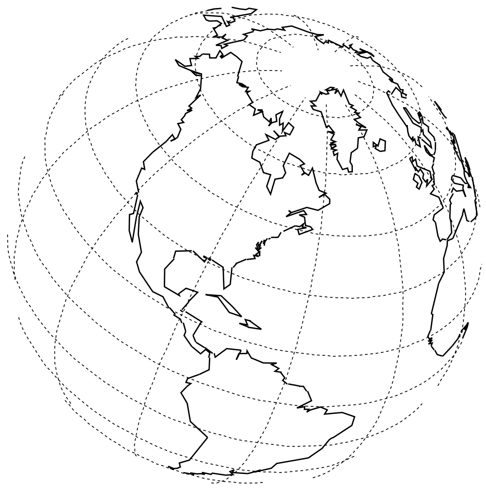
The GIL Graphical Interface Builder



User's Manual, Version 1.0

Project **GeoSim**
John B. Raley
Department of Computer Science
Virginia Tech
Blacksburg, VA 24061
email: geosim@cs.vt.edu

Contents

1 Introduction

GIL, the **GeoSim Interface Library**, is a set of functions and tools for implementing a graphical user interface (GUI). Programming with **GIL** is covered in the **GIL** *Programmer's Manual.* The **GIL Graphical Interface Builder** is a tool for developing user interfaces for **GIL** applications. **Graphical Interface Builder** is a *WYSIWYG* (what-you-see-is-what-you-get) editor for **GIL** interfaces. **Graphical Interface Builder** displays interfaces as they will appear in the application, and gives the developer an intuitive way to develop interfaces.

GIL interfaces are stored in text files called *interface files.* It is possible to hand-code an interface by writing an interface file in a text editor. However, interface development will proceed more quickly when the **Graphical Interface Builder** is employed. Because **Graphical Interface Builder** allows the developer to see the interface as it will appear in the application, the developer can make changes to the interface, then evaluate those changes, without having to re-run the application. Also, interface files generated with **Graphical Interface Builder** are free from syntax errors which inevitably occur in hand-coded interface files.

Graphical Interface Builder resembles a simple drawing package in many ways. The user may "draw" new interface elements, "drag" existing elements to new locations, and edit an element's properties by clicking on the element. Also, **Graphical Interface Builder** can be used to edit interface parameters, such as global button characteristics and the color palette. Additionally, **Graphical Interface Builder** allows the user to create, edit, and remove **GIL** functions, color aliases, and messages.

Before reading this manual, you should be familiar with **GIL** interface files. **GIL** interface files are covered in Section 4 of the **GIL** *Programmer's Manual.*

1.1 Creating and Editing Interfaces

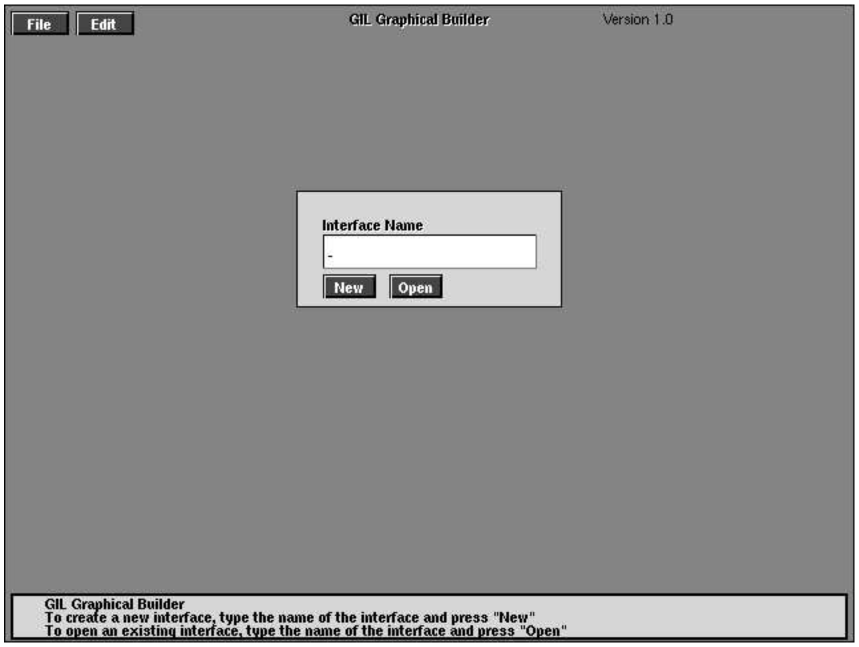
When **Graphical Interface Builder** starts, a window similar to the one in Figure 1 will be displayed. The white box under the label `Interface Name` is an **edit box**. When the text in the box is drawn in black, the edit box has **focus**. When an edit box has focus, typing is sent to the edit box. Only one edit box at a time can have focus. To give an edit box focus, click the mouse in the edit box. Characters are inserted at the **insertion point**, indicated by an underscore character. The insertion point can be adjusted by the left and right arrow keys; pressing the `delete` key removes the character to the left of the insertion point.

To open an existing interface, type the name of the interface file into the `Interface Name` edit box, and press the **'Open'** button. To create a new interface, type the name of the new interface and press the **'New'** button.

Once the interface has been opened or created, the **visual editor** will be displayed (see Section 3).

1.2 Saving, Reverting, and Closing Interfaces; Quitting Graphical Interface Builder

To save the current version of the interface to disk, select **'Save'** from the **'File'** menu. This writes the current version of the interface to disk.



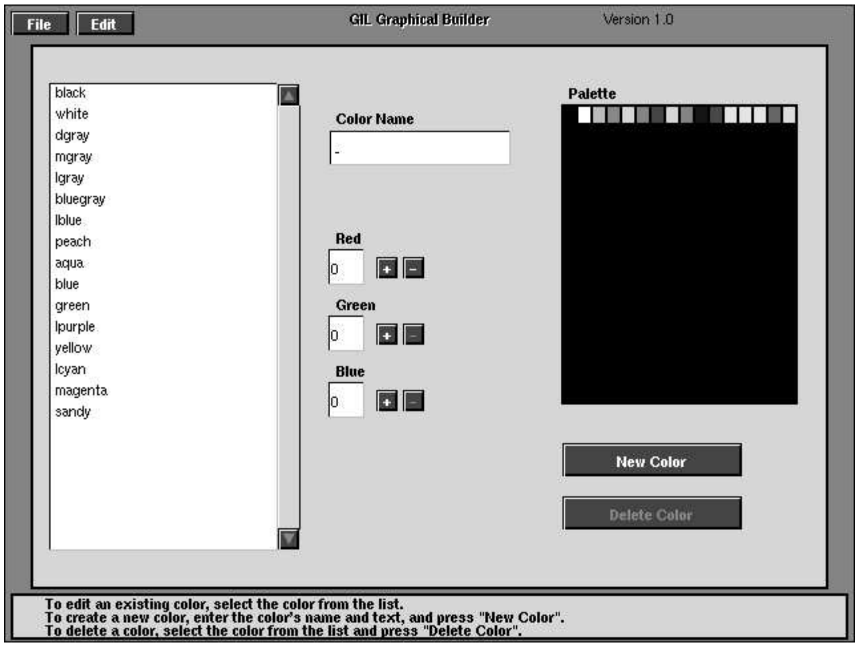
GIL Graphical Builder
To create a new interface, type the name of the interface and press "New"
To open an existing interface, type the name of the interface and press "Open"

Figure 1: **Graphical Interface Builder** Startup Window

Occasionally, the user may want to "throw away" the latest set of changes to an interface, and revert back to the latest saved version of the interface. Selecting **'Revert'** from the **'File'** menu causes the latest saved version of the interface to be read into **Graphical Interface Builder**.

The user should select **'Close Interface'** from the **'File'** menu when he or she is finished editing the current interface, and wants to edit another interface. Before closing and interface, **Graphical Interface Builder** will give the user the option to save the current version of the interface.

To close the current interface and quit the **Graphical Interface Builder**, select **'Quit'** from the **'File'** menu. The user will be given the option to save the interface before quitting.



To edit an existing color, select the color from the list.
To create a new color, enter the color's name and text, and press "New Color".
To delete a color, select the color from the list and press "Delete Color".

Figure 2: Color Palette Window

2 Interface Parameters and Non-Visual Elements

This section describes how to use **Graphical Interface Builder** to edit interface parameters and non-visual interface elements (functions, color aliases, and messages).

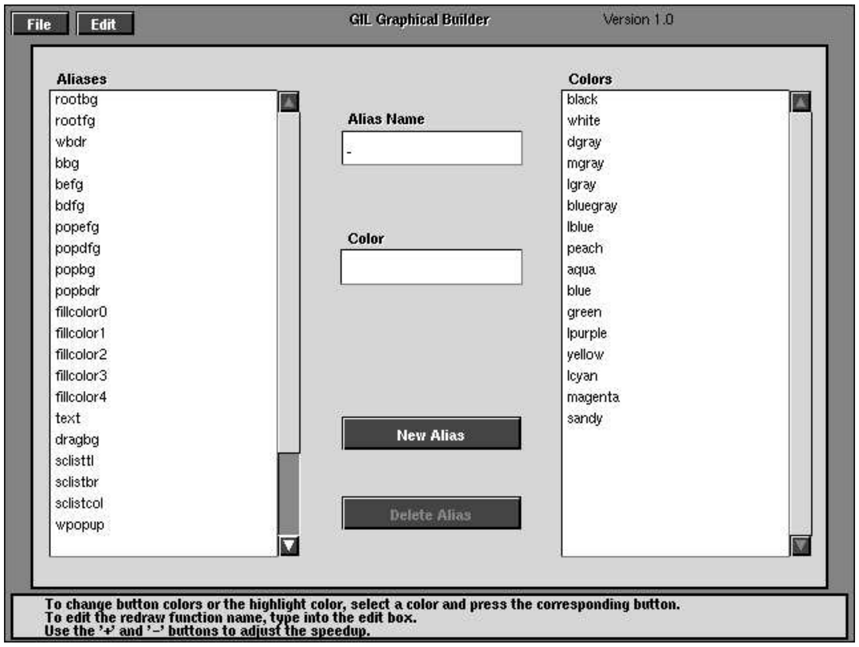
Returning to the visual editor The visual editor can be invoked by selecting **'Interface'** from the **'Edit'** menu.

2.1 Color Palette

The **color palette** defines the specific colors available for use in the interface. Up to 256 colors can be included in the color palette. Each color in the color palette is specified by a name and a set of three values called Red-Green-Blue (RGB) values. Each RGB value must be in the range 0-63, inclusive.

To edit the color palette of an interface, select **'Palette'** from the **'Edit'** menu. A window similar to the one in Figure 2 will be displayed.

The names of colors in the color palette are displayed in the scrolling list on the left side of the window. The colors are displayed in the black rectangle, under the label `Palette`, on the right of the window. To edit a color, select the color from the list, or click the mouse on the color in the `Palette` rectange. The color's name will appear in the edit box under the label `Color Name`. The color's red, green, and blue values will be displayed in the boxes under the labels `Red`, `Green`, `Blue`, respectively. The color's name can be edited in the edit box. To increase a component value of the color, press the '+' button next to the component. To decrease a value, press the '–' button next



To change button colors or the highlight color, select a color and press the corresponding button.
To edit the redraw function name, type into the edit box.
Use the '+' and '–' buttons to adjust the speedup.

Figure 3: Color Alias Window

to the component. The new color will appear in the `Palette` label when another color is selected.

New colors can be created when no existing color is selected. To create a new color, enter the color's name in the `Color Name` edit box, and adjust the red, green, and blue components of the color. Then, press the **'New Color'** button. The new color name will appear in the scrolling list, and the new color will appear in the `Palette` rectangle. NOTE: New colors cannot be created while an existing color is selected. To unselect an existing color, press the `New Color` button.

To delete an existing color, select the color and press the **'Delete Color'** button.

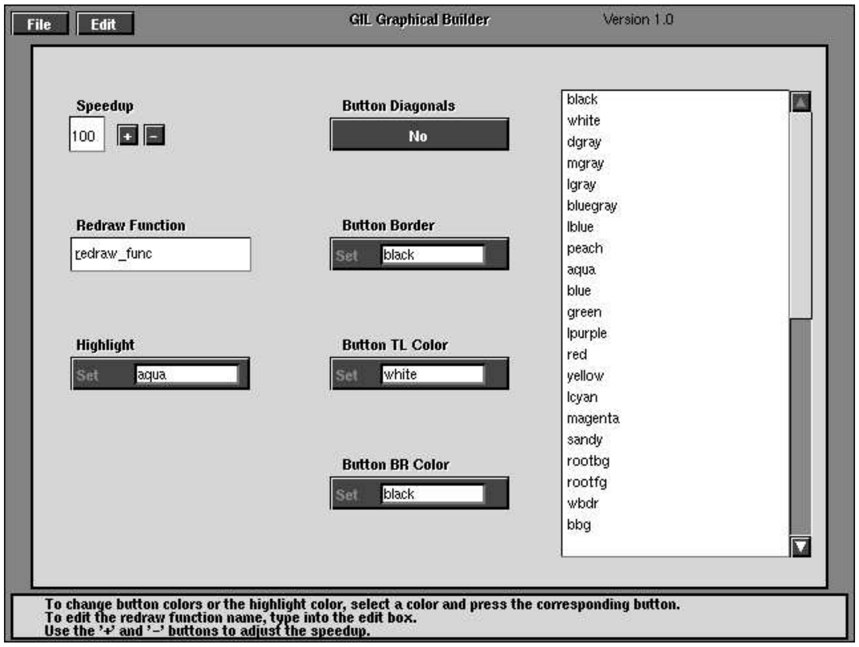
2.2 Color Aliases

In a typical **GIL** interface, color names in the color palette describe the appearance of the colors, not the ways in which these colors are used. It is better to indicate color use with a **color alias**. A color alias is "translated" to a color when the application starts. Color aliases make it easier to modify the colors used in an interface.

To edit the colors aliases in an interface, select **'Color Aliases'** from the **'Edit'** menu. A window similar to the one in Figure 3 will be displayed.

The names of existing aliases are displayed in the scrolling list on the left side of the window, under the label `aliases`. The colors in the color palette are listed on the right of the window, under the label `Colors`. To edit a color alias, select the alias from the alias list. The alias name will be displayed in the edit box under the label `Alias Name`. The color which the alias references will be displayed in the white box under the label `Color`. The color alias's name can be edited in the edit box. To change which color an alias references, select the new color from the color list.

New aliases can be created when no existing alias is selected. To create a new alias, enter the alias's



Speedup: 100

Redraw Function: redraw_func

Highlight: aqua

Button Diagonals: No

Button Border: black

Button TL Color: white

Button BR Color: black

black
white
dgray
mgray
lgray
bluegray
lblue
peach
aqua
blue
green
lpurple
red
yellow
lcyan
magenta
sandy
rootbg
rootfg
wbdr
bbg

To change button colors or the highlight color, select a color and press the corresponding button.
To edit the redraw function name, type into the edit box.
Use the '+' and '−' buttons to adjust the speedup.

Figure 4: Interface Parameters Window

name in the `Alias Name` edit box. Choose a color from the color list to assign to this alias. Then, press the **'New Alias'** button. NOTE: New aliases cannot be created while an existing alias is selected. To unselect an existing alias, press the **'New Alias'** button.

To delete an existing alias, select the alias from the alias list, and press the **'Delete Alias'** button.

2.3 Interface Parameters

Interface parameter declarations specify properties which are global to the entire interface. Interface parameters include:

- The acceleration rate of repeatable buttons.
- The name of the redraw function.
- The color used to highlight the current selection on a popup menu.
- Global button parameters, including edge and border colors, and whether to draw a diagonal line across disabled buttons.

To edit the parameters of an interface, select **'Parameters'** from the **'Edit'** menu. A window similar to the one in Figure 4 will be displayed.

The interface's speedup – the acceleration rate of repeatable buttons – is displayed in the field under the `Speedup` label. To increase the speedup, press the '+' button beside the field where the speedup is displayed. To decrease the speedup, press the '−' button.

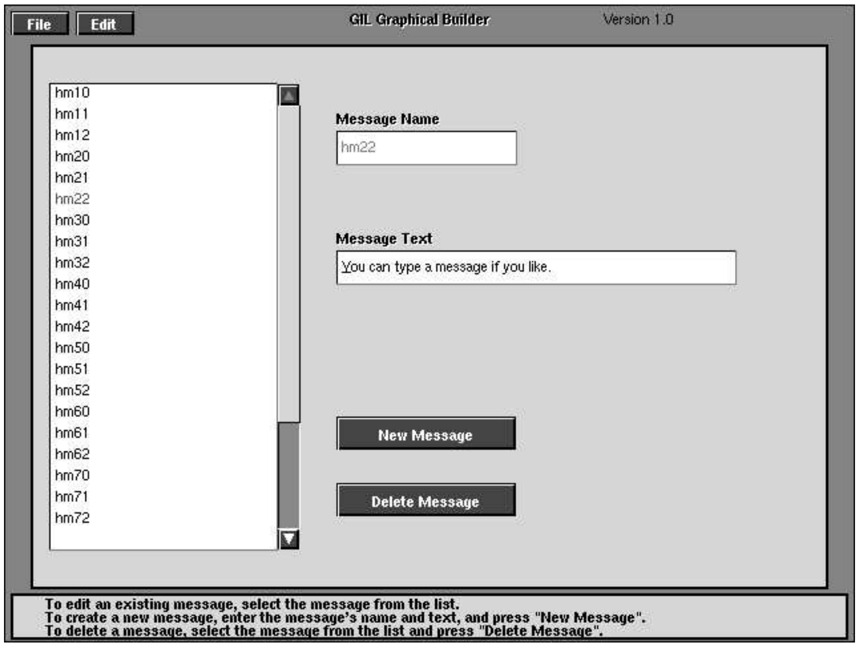
The redraw function name is displayed in the box under the `Redraw Function` label. The box which displays the redraw function name is an **edit box**.

The popup menu highlight color name is shown in a white field inside of the button under the `Highlight` label. To change the highlight color, select a color in the color list at the right of the window, and press the button which displays the highlight color. The selected color will become the highlight color.

The button under the `Button Diagonals` label controls whether a disabled button will have a diagonal line drawn through it. To toggle this setting, press the button.

The three button color parameters – border color, top-left color, and bottom-right color – are controled by the buttons under the `Button Border`, `Button TL Color`, and `Button BR Color` labels, respectively. To change one of these parameters, select the desired color or alias from the color picklist, then press the button corresponding to the parameter you wish to change.

2.4 Messages



Message Name

hm22

Message Text

You can type a message if you like.

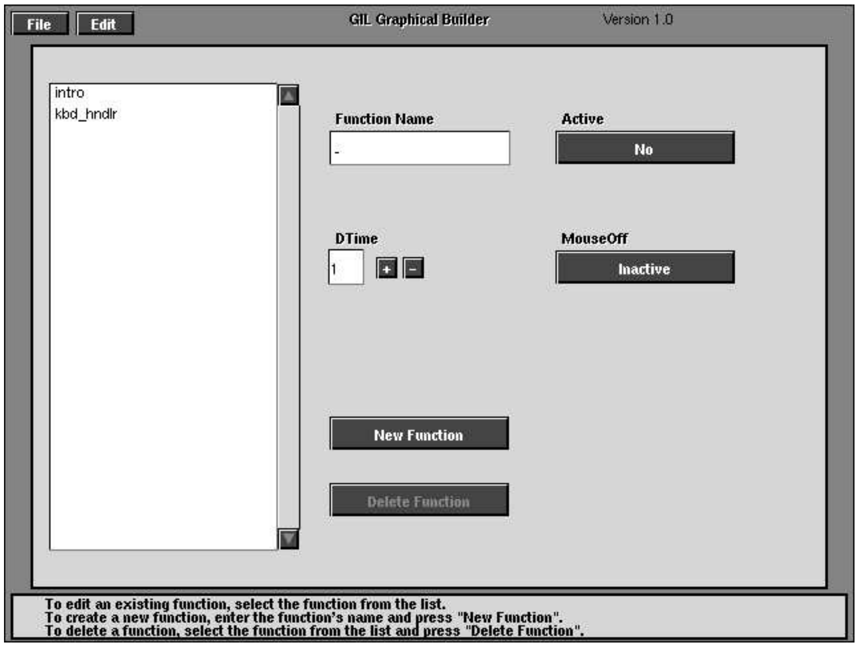
To edit an existing message, select the message from the list.
To create a new message, enter the message's name and text, and press "New Message".
To delete a message, select the message from the list and press "Delete Message".

Figure 5: Messages Window

GIL provides support for displaying help messages to the user. Messages may be displayed in the interface window named `message` with the `GSdisplayhelpmessage` function, or in the popup window `alert` with the GSalert function. Messages must be declared in the interface in order to be displayed.

To edit the messages in an interface, select **'Messages'** from the **'Edit'** menu. A window similar to the one in Figure 5 will be displayed.

The names of messages in the interface are displayed in the scrolling list on the left side of the window. To edit a message, select the message from the list. The message's name will be displayed



intro
kbd_hndlr

Function Name: -

Active: No

DTime: 1 + –

MouseOff: Inactive

To edit an existing function, select the function from the list.
To create a new function, enter the function's name and press "New Function".
To delete a function, select the function from the list and press "Delete Function".

Figure 6: Independent Functions Window

in the white box under the label `Message Name`. The message's text will be displayed in the edit box under the label `Message Text`.

New messages can be created when no existing message is selected. To create a new message, enter the new message's name in the `Message Name` box, and its text in the `Message Text` box. Then press the **'New Message'** button. NOTE: New messages cannot be created while an existing message is selected. To unselect an existing message, press the **'New Message'** button.

To delete an existing message, select the message in the message list, and press the **'Delete Message'** button.

2.5 Functions

A **GIL** application can execute functions which are not associated with an interface element. Such functions are called **independent functions**. **GIL** executes active independent function each time it iterates the event loop. Independent functions must be declared in the interface.

To edit the independent functions of an interface, select **'Functions'** from the **'Edit'** menu. A window similar to the one in Figure 6 will be displayed.

The names of declared functions are displayed in the scrolling list on the left side of the window. To edit a function, select the function from the list. The function's name will be displayed in the edit box under the label `Function Name`. The function's **dtime**, or time between executions, will be displayed in the white box under the label `dtime`. The function name can be edited in the edit box. To increase the time between executions, press the '+' button. To decrease this time, press the button '–' button.

The button under the label `Active` controls whether the function is initially active. The button displays the current setting. To toggle this setting, press the button. The button under the label `MouseOff` controls whether the mouse is turned off when this function executes. This setting is meaningful only when the application runs under MS-DOS.

New functions can be created when no existing function is selected. To create a new function, enter the new function's name in the `Function Name` edit box, and set its active status and **dtime** – time between calls. Then press the **'New Function'** button. NOTE: New functions cannot be created while an existing function is selected. To unselect an existing function, press the **'New Function'** button.

To delete an existing function, select the function from the function list, and press the **'Delete Function'** button.

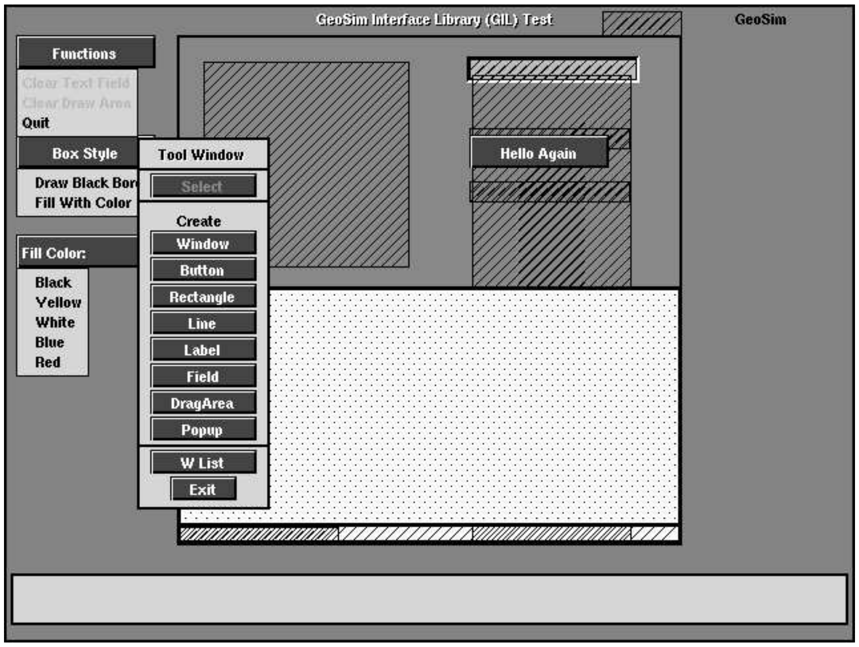


Figure 7: The *giltest* Interface

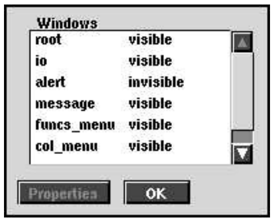
3 Interface Elements

When an interface is opened in the **Graphical Interface Builder**, the interface is displayed in the **visual editor**. The visual editor enables the user to "draw" new interface elements, "drag" existing interface elements to new locations, remove interface elements, and change the properties of interface elements.

Property Windows Every element in a **GIL** interface has certain properties. For example, the properties for a dragarea include its name, dimensions, color, function, and active status. When an interface element is created or selected, a window showing the element's properties is displayed. This window is a **property window**. Property windows enable the user to edit the element's properties.

All property windows have **'Cancel'**, **'Delete'**, and **'OK'** buttons. Pressing **'Cancel'** removes the property window and restores the interface element to its original state. If the property window was displayed because an element was created, the element is deleted. Pressing **'Delete'** deletes the element and removes the property window. Pressing **'OK'** updates the element's properties and removes the property window.

The Visual Editor Toolbar The visual editor is controled by its toolbar. The toolbar "floats" above the interface in the visual editor, and can be dragged to a new location by pressing the mouse button while the mouse is in the tool window (but not on a button), moving the mouse, and releasing the mouse button. Also, the tool window can be hidden by pressing any key. To bring back the tool window when it is hidden, press any key.



Windows

root	visible
io	visible
alert	invisible
message	visible
funcs_menu	visible
col_menu	visible

Figure 8: Window List

The first nine buttons in the tool window control the mode of the visual editor. The first button, **'Select'**, puts the visual editor in **select mode**. In this mode, existing elements can be dragged to new locations, and the properties of existing elements can be edited. The buttons under the label `Create` put the visual editor in **create mode**. In this mode, new elements can be created. The type of elements created depends on which create button was pressed. For a complete discussion of select mode and create mode, see Section 3.1.

The button **'W List'** displays a window similar to the one shown in Figure 8. The scrolling list in this window contains all windows and popup menus in the interface. Some windows declared in the interface may not be shown in the visual editor. These windows have `invisible` beside their names in the list. Windows which are visible in the visual editor have `visible` beside their names. Initially, windows which are declared inactive are not shown in the visual editor. However, whether a window is visible in the visual editor does not affect whether it is active in the interface. To toggle a window's visibility, click the mouse on its `visible / invisible` label. To display a window's properties, select the window in the list and press the **'Properties'** button. To remove the window list, press the **'OK'** button.

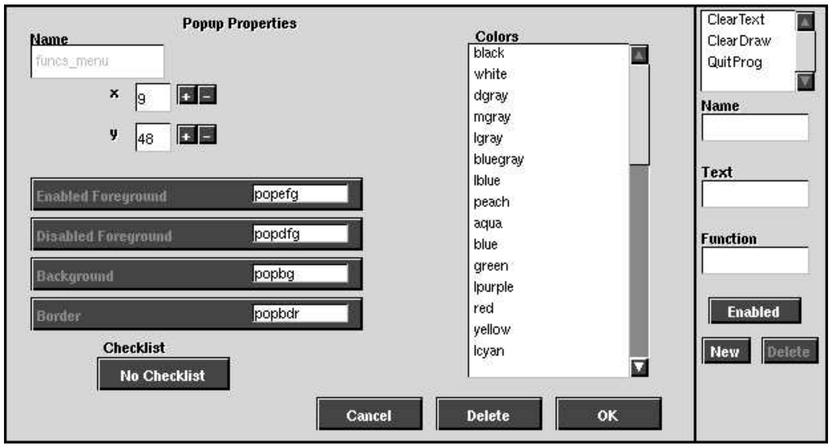
The button **'Exit'** ends visual mode editing, and displays the Interface Parameters window (see Section 2.3).

3.1 Visual Editor Operation Modes

The Visual Editor operates primarily in two modes: select mode and create mode. In select mode, the user can "drag" an object by positioning the mouse over the object, holding down the mouse button, moving the mouse to a new location, and releasing the mouse button. Also, the user can select an object by clicking the mouse on the object. This causes the object's property window to be displayed.

When the visual editor is in create mode, new elements can be created. To create a new object, position the mouse over one corner of the object, and press the mouse button. Move the mouse to the opposite corner of the object. Notice the outline of the object, which follows the mouse movement. Release the mouse button. The property window of the new object will be displayed.

There are two exceptions to the create method: popup menus and labels. To create popup menus and labels, simply click the mouse on their desired location. The dimensions of popup menus and labels cannot be explicitly specified (their size is controlled by the text that forms them).



Name: funcs_menu
x: 9
y: 48
Enabled Foreground: popefg
Disabled Foreground: popdfg
Background: popbg
Border: popbdr
Checklist: No Checklist

Figure 9: Popup Menu Property Window

3.2 Popup Menu Properties

The popup property window, shown in Figure 9, is displayed after a popup menu has been created or selected.

The popup menu name is shown in the edit box under the label `Name`. To edit the popup menu name, focus the edit box by clicking in it and type the new name.

The coordinates of the top-left corner of the popup menu are shown in the white fields beside the labels `x` and `y`. To increase one of the coordinates, press the '+' button beside the coordinate. To decrease a coordinate, press the '−' button.

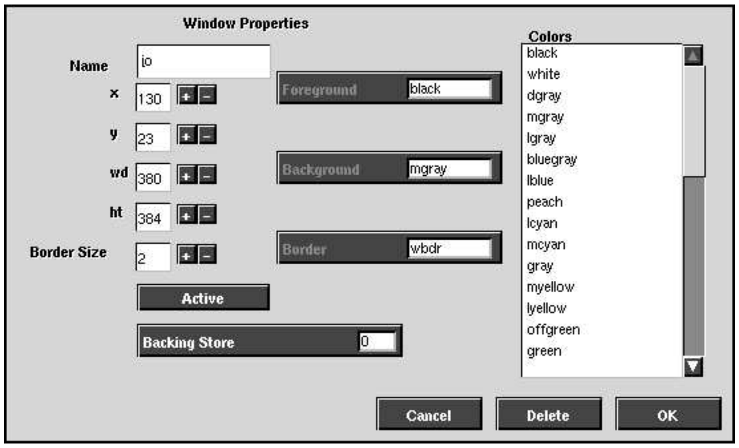
The four popup menu color properties — enabled foreground color, disabled foreground color, background color, and border color — are displayed in the buttons with these names. To change a color property, select a color from the color list, and press the button corresponding to the property that will change. The color property will change to the selected color.

Some popup menus allow "checked" items. The button under the `Checklist` label controls whether the popup menu allows checked items. Press the button to toggle this setting.

On the right side of the popup menu property window is a scrolling list of menu items – items which will appear in the popup menu. To edit a menu item, select the item from the list. The item's name will be displayed in the edit box under the label `Name`. The item's text will appear in the edit box labeled `Text`, and the item's function will appear in the edit box labeled `Function`. The button under the `Function` edit box will read `Enabled` if the item is enabled, and `Disabled` if the item is not enabled. The item name, text, and function can be edited in the edit boxes. Press the `Enabled` button to toggle this setting.

New menu items can be created when no existing menu item is selected. To create a new menu item, enter the item's name, text, and function, set its `enabled` status, and press the **'New'** button. NOTE: New menu items cannot be created while an existing menu item is selected. To unselect an existing menu item, press the **'New'** button.

To delete a menu item, select the item from the item list, and press the **'Delete'** button.



Name io
x 130
y 23
wd 380
ht 384
Border Size 2
Foreground black
Background mgray
Border wbdr
Backing Store 0

Figure 10: Window Property Window

When you are finished editing the popup menu's properties, press the **'OK'** button. To cancel changes to the popup menu, press **'Cancel'**. To delete the popup menu, press **'Delete'**. If the popup menu property window was displayed because a new popup menu was created, pressing **'Cancel'** deletes the popup menu.

3.3 Window Properties

The window property window, shown in Figure 10, is displayed after a window has been created or selected.

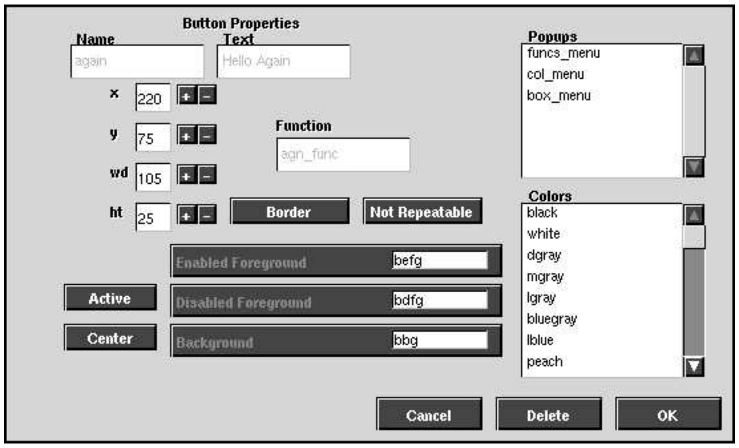
The window's name is shown in the white edit box to the right of the label `Name`. To edit the window name, focus the edit box and type the new name.

The window's origin and dimensions – its top-left corner and its width and height – are shown in the fields to the right of the labels `x`, `y`, `wd`, and `ht`, respectively. To increase a value, press the '+' button beside the value. To decrease a value, press the '–' button. The window's border size is shown below the window's height. To increase the border size, press the '+' button beside the border size; to decrease the border size, press the '–' button.

The window's color properties – foreground color, background color, and border color – are displayed in the buttons with these names. To change a color property, select a color from the color list, and press the button corresponding to the property that will change. The property will change to the selected color.

A window can be declared active or inactive. An active window is visible when the application starts up; an inactive window is not visible at start-up. The button under the border size field displays the window's active status. Press this button to toggle the active status.

Backing store can be used to save a portion of the screen which is overdrawn by a **GIL** window. When the window is removed, the old contents of the screen can be copied back onto the screen, avoiding redrawing the screen from scratch. There are several backing store areas available to the programmer. To allow a window to use backing store, press the **'Backing Store'** button, and select the desired backing store from the popup menu. Note that popup menus use backing store



Name: again
Text: Hello Again
x: 220
y: 75
wd: 105
ht: 25
Function: agn_func
Enabled Foreground: befg
Disabled Foreground: bdfg
Background: bbg
Popups: funcs_menu, col_menu, box_menu
Colors: black, white, dgray, mgray, lgray, bluegray, lblue, peach

Figure 11: Button Property Window

1; if a window which uses backing store can be visible when a popup menu is displayed, it must use another backing store.

When you are finished editing the window's properties, press the **'OK'** button. To cancel changes to the window, press **'Cancel'**. To delete the window, press **'Delete'**. If the window property window was displayed because a window was created, pressing **'Cancel'** deletes the window.

3.4 Button Properties

The button property window, shown in Figure 11, is displayed after a button has been created or selected.

The button's name is shown in the white edit box under the label `Name`. The button's text is in the edit box under the label `Text`, and the button's function is in the edit box under the label `Function`. To edit any of these properties, focus the edit box and type the new value. A button's function can associate the button with a popup menu. The popup menus in the interface are displayed in the picklist labeled `Popups`. To associate the button with a popup menu, select a popup menu from the list. The menu name will be copied into the `Function` edit box, preceded by the '`@`' character.

The button's origin and dimensions – its top-left corner and its width and height – are shown in the fields to the right of the labels `x`, `y`, `wd`, and `ht`, respectively. To increase any of these values, press the '+' button beside the value. To decrease a value, press the '–' button.

The button's color properties — enabled foreground, disabled foreground, and background — are displayed in the buttons with these names. To change a color property, select a color from the color list, and press the button corresponding to the property that will change. The property will change to the selected color.

Buttons can be drawn with or without borders. The button immediately to the right of the height field displays the current setting. To toggle this setting, press the button.

Buttons can be repeatable. The function associated with a repeatable buttons is iterated as long as the button is pressed. The button **'Repeatable'**/**'Not Repeatable'** displays the current setting. Press the button to toggle this setting.

The button to the left of the **'Disabled Foreground'** button displays the button's active status: whether the button is enabled, disabled, or invisible at start-up time. To change this setting, press the button and select the new setting from the popup menu.

The button to the left of the **'Background'** button displays the button's text justification. To change the text justification, press the button and select the new setting from the popup menu.

When you are finished editing the button's properties, press the **'OK'** button. To cancel changes to the button, press **'Cancel'**. To delete the button, press **'Delete'**. If the button property window was displayed because a button was created, pressing **'Cancel'** deletes the button.

3.5 Field Properties

The field property window, shown in Figure 12, is displayed after a field has been created or selected.

The field's name is shown in the white edit box to the right of the label `Name`. To edit the field's name, focus the edit box by clicking in it, and type the new name.

The field's origin and dimensions – its top-left corner and its width and height – are shown in the fields to the right of the labels `x`, `y`, `wd`, and `ht`, respectively. To increase a value, press the '+' button beside the value. To decrease a value, press the '–' button.

The field's color is displayed in the **'Color'** button. To change the field's color, select a color from the color list, and press the **'Color'** button. The field's color will change to the selected color.

When you are finished editing the field's properties, press the **'OK'** button. To cancel changes to the field, press **'Cancel'**. To delete the field, press **'Delete'**. If the field property window was displayed because a field was created, pressing **'Cancel'** deletes the field.

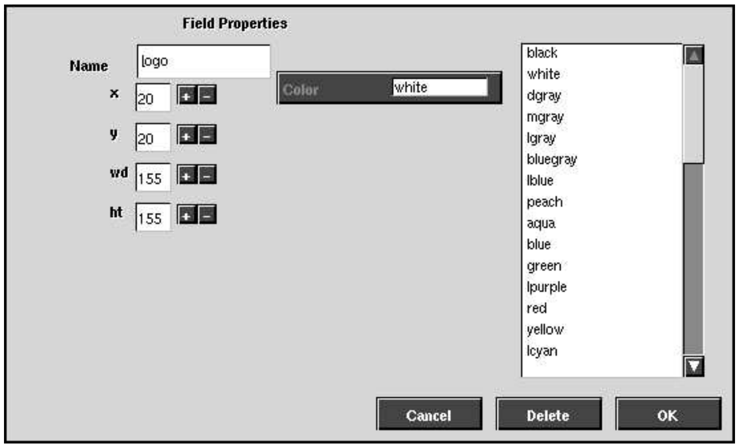
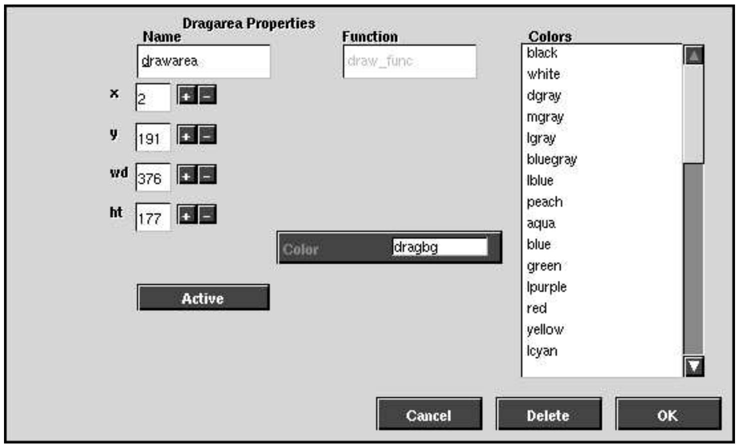


Figure 12: Field Property Window


Dragarea Properties

Name: drawarea

Function: draw_func

x: 2

y: 191

wd: 376

ht: 177

Color: dragbg

Colors: black, white, dgray, mgray, lgray, bluegray, lblue, peach, aqua, blue, green, lpurple, red, yellow, lcyan

Figure 13: Dragarea Property Window

3.6 Dragarea Properties

The dragarea property window, shown in Figure 13, is displayed after a dragarea has been created or selected.

The dragarea's name is shown in the edit box to the right of the label `Name`. The dragarea's function is in the edit box under the label `Function`. To edit one of these properties, click in the edit box, and type the new value.

The dragarea's origin and dimensions – its top-left corner and its width and height – are shown in the fields to the right of the labels `x`, `y`, `wd`, and `ht`, respectively. To increase a value, press the '+' button beside the value. To decrease a value, press the '−' button.

The dragarea's color is displayed in the **'Color'** button. To change the dragarea's color, select a color from the color list, and press the **'Color'** button. The dragarea's color will change to the selected color.

A dragarea can be active or inactive. The active status of the dragarea is shown in the button under the `ht` field. To toggle this setting, press the **'Active'**/**'Inactive'** button.

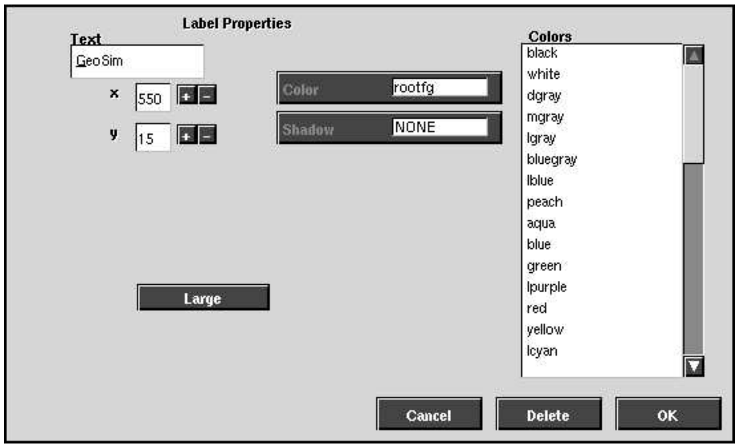
When you are finished editing the dragarea's properties, press the **'OK'** button. To cancel changes to the dragarea, press **'Cancel'**. To delete the dragarea, press **'Delete'**. If the dragarea property window was displayed because a dragarea was created, pressing **'Cancel'** deletes the dragarea.

3.7 Label Properties

The label property window, shown in Figure 14, is displayed after a label has been created or selected.

The label's text is shown in the edit box under the label `Text`. To edit the text, click in the edit box, and type the new text.

The label's origin is shown in the fields beside the labels `x` and `y`. The bottom left corner of the label is at these coordinates. To change a coordinate, press the '+' or '−' button beside the


Label Properties

Text: GeoSim

x 550

y 15

Color rootfg

Shadow NONE

Figure 14: Label Property Window

coordinate.

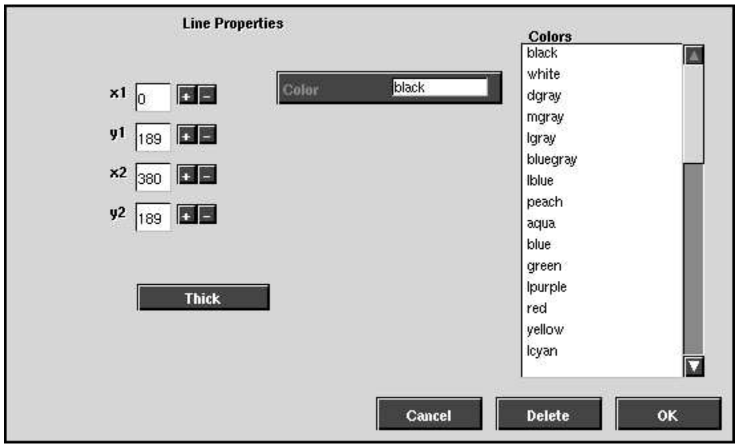
The color shown in the **'Color'** button is the color used to draw the label's text. The text is shadowed with the color in the **'Shadow'** button. The shadow color can be `NONE`, indicating that no shadow color is to be used. To change either the color or shadow color, select a color from the color list, and press the button corresponding to the property which will change.

A label's text can be large or small. The label text size is shown in the button under the **y** field. Press the button to toggle this setting.

When you are finished editing the label's properties, press the **'OK'** button. To cancel changes to the label, press **'Cancel'**. To delete the label, press **'Delete'**. If the label property window was displayed because a label was created, pressing **'Cancel'** deletes the label.

3.8 Line Properties

The line property window, shown in Figure 15, is displayed after a line has been created or selected.


Line Properties

x1 0

y1 189

x2 380

y2 189

Color black

Figure 15: Line Property Window

The line's endpoints are shown in the fields to the right of the labels `x1`, `y1`, `x2`, and `y2`. To change a coordinate, press the '+' or '–' button beside the coordinate.

The color shown in the **'Color'** button is the line's color. To change the line's color, select the new color from the color list, and press the **'Color'** button.

A line can be thin (1 pixel wide) or thick (2 pixels wide). The button under the `y2` label displays this property. To toggle this property, press the button.

When you are finished editing the line's properties, press the **'OK'** button. To cancel changes to the line, press **'Cancel'**. To delete the line, press **'Delete'**. If the line property window was displayed because a line was created, pressing **'Cancel'** deletes the line.

3.9 Rectangle Properties

The rectangle property window, shown in Figure 16, is displayed after a rectangle has been created or selected.

The rectangles's origin and dimensions – its top-left corner and its width and height – are shown in the fields to the right of the labels `x`, `y`, `wd`, and `ht`, respectively. To increase a value, press the '+' button beside the value. To decrease a value, press the '–' button.

The color shown in the **'Color'** button is the rectangle's color. To change the rectangle's color, select the new color from the color list, and press the **'Color'** button.

Rectanges can be filled, or hollow. In a filled rectangle, all interior points of the rectangle are shown in the rectangle's color. To toggle this setting, press the **'Fill'**/**'No Fill'** button.

A rectangle's border can be thin (1 pixel wide) or thick (2 pixels wide). The button under the `y2` label displays this property. To toggle this property, press the button. Note that if the rectangle is filled, this property does not affect the rectangle's appearance.

When you are finished editing the rectangle's properties, press the **'OK'** button. To cancel changes to the rectangle, press **'Cancel'**. To delete the rectangle, press **'Delete'**. If the rectangle property window was displayed because a rectangle was created, pressing **'Cancel'** deletes the rectangle.

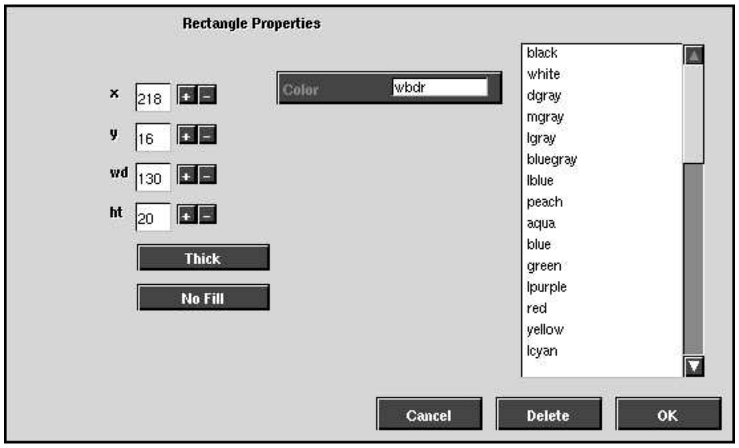


Figure 16: Rectangle Property Window

4 Configuring Graphical Interface Builder

This section describes how to set **Graphical Interface Builder**'s options.

When **Graphical Interface Builder** starts, it looks in the current directory for `gbuilder.gcf`, its **configuration file**. The configuration file specifies **Graphical Interface Builder**'s options and default entries in new interface elements. The fields in `gbuilder.gcf` are:

`interfacefile` - **Graphical Interface Builder**'s own interface file.

`defaultwforeground` - the default window foreground color.

`defaultwbackground` - the default window background color.

`defaultwbordercol` - the default window border color.

`showpropaftercreate` - `TRUE` to display property windows for newly created objects; `FALSE` otherwise.

`defaultbefg` - default button enabled foreground color.

`defaultdefg` - default button enabled background color.

`defaultbbg` - default button border color.

`defaultdragcol` - default dragarea color.

`defaultfieldcol` - default field color.

`defaultpopefg` - default popup menu enabled foreground color.

`defaultpopdfg` - default popup menu disabled foreground color.

`defaultpopbg` - default popup menu background color.

`defaultpopbdr` - default popup border color.

Each field in `gbuilder.gcf` must appear on a separate line. The format for a line is: the field name, followed by `'= '`, followed by the field value.