Chapter 9

Customizing Prism

This chapter discusses ways in which you can change various aspects of Prism's appearance and the way Prism operates. To learn:

- **How to use the tear-off region,** see Section 9.1, below.
- **How to set up alternative names for commands and variables,** see Section 9.2.
- **How to change Prism defaults by using the Customize utility,** see Section 9.3.
- **How to change Prism defaults in your X resource database,** see Section 9.4.
- **How to initialize Prism,** see Section 9.5.

9.1 Using the Tear-Off Region

You can place frequently used menu selections and commands in the tear-off region below the menu bar; in the tear-off region, they become buttons that you can click on to execute them. Figure 37 shows the buttons that are there by default.

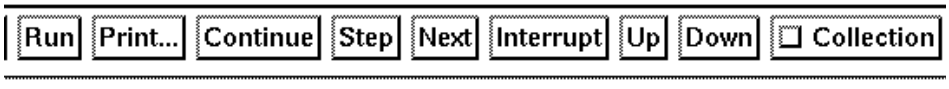

Figure 37. The tear-off region.

Copyright © 1991–1994 Thinking Machines Corporation

Putting menu selections and commands in the tear-off region lets you get access to them without having to pull down a menu or issue a command from the command line.

Changes you make to the tear-off region are saved when you leave Prism; see Section 9.3.3.

9.1.1 Adding Menu Selections to the Tear-Off Region

From the menu bar: To add a menu selection to the tear-off region, first enter tear-off mode by choosing **Tear-off** from the **Utilities** menu. A dialog box appears that describes tear-off mode; see Figure 38.

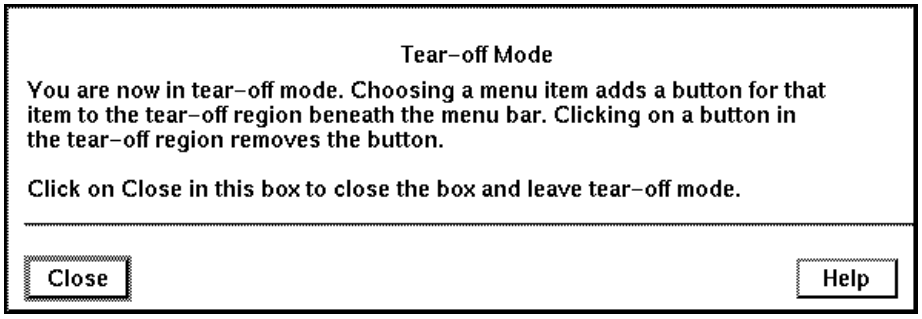

Figure 38. The Tear-Off dialog box.

While the dialog box is on the screen, choosing any selection from a menu adds a button for this selection to the tear-off region. Clicking on a button in the tear-off region removes that button. If you fill up the region, you can resize it to accommodate more buttons. To resize the region, drag the small resize box at the bottom right of the region.

Click on **Close** or press the **Esc** key while the mouse pointer is in the dialog box to close the box and leave tear-off mode.

When you are not in tear-off mode, clicking on a button in the tear-off region has the same effect as choosing the equivalent selection from a menu.

From the command window: Use the **`tearoff`** and **`untearoff`** commands from the command window to add menu selections to and remove them from the tear-off region. Put the selection name in quotation marks; case

Version 2.0, April 1994
Copyright © 1991–1994 Thinking Machines Corporation

doesn't matter, and you can omit spaces and the ellipsis (...) that indicates the selection displays a window or dialog box. If the selection name is ambiguous, put the menu name in parentheses after the selection name. For example,

```
tearoff "print (events)"
```

adds a button for the **Print** selection from the **Events** menu to the tear-off region.

9.1.2 Adding Prism Commands to the Tear-Off Region

To add a Prism command to the tear-off region, issue the **pushbutton** command, specifying the label for the tear-off button and the command it is to execute. The label must be a single word. The command can be any valid Prism command, along with its arguments. For example,

```
pushbutton printa print a on dedicated
```

adds a button labeled **printa** to the tear-off region. Clicking on it executes the command **print a on dedicated**.

To remove a button created via the **pushbutton** command, you can either click on it while in tear-off mode, or issue the **untearoff** command as described above.

9.2 Setting Up Alternative Names for Commands and Variables

Prism provides commands that let you create alternative names for commands, variables, and expressions.

Use the **alias** command to set up an alternative name for a Prism command. For example,

```
alias ni nexti
```

makes **ni** an alias for the **nexti** command. Prism provides some default aliases for common commands. Issue **alias** with no arguments to display

Copyright © 1991–1994 Thinking Machines Corporation

a list of the current aliases. Issue the **unalias** command to remove an alias. For example,

```
unalias ni
```

removes the alias created above.

Use the **set** command to set up an alternative name for a variable or expression. For example,

```
set alan = annoyingly_long_array_name
```

abbreviates the annoyingly long array name to **alan**. You can use this abbreviation subsequently in your program to refer to this variable. Use the **unset** command to remove a setting. For example,

```
unset alan
```

removes the setting created above.

Changes you make via **alias** and **set** last for your current Prism session. To make them permanent, you can add the appropriate commands to your **.prisminit** file; see Section 9.5.

9.3 Using the Customize Utility

Many aspects of Prism's behavior and appearance — for example, the colors it displays on color workstations, and the fonts it uses for text — are controlled by the settings of *Prism resources*. The default settings for many of these resources appear in the file **Prism** in the X11 **app-defaults** directory for your system. Your system administrator can change these system-wide defaults. You can override these defaults in two ways:

- For many of them, you can use the **Customize** selection from the **Utilities** menu to display a window in which you can change the settings. This section describes this method.
- A more general method is to add an entry for a resource to your X resource database, as described in the next section. Using the **Customize** utility is, however, much more convenient.

Version 2.0, April 1994
Copyright © 1991–1994 Thinking Machines Corporation

Choosing **Customize** from the **Utilities** menu displays the window shown in Figure 39.

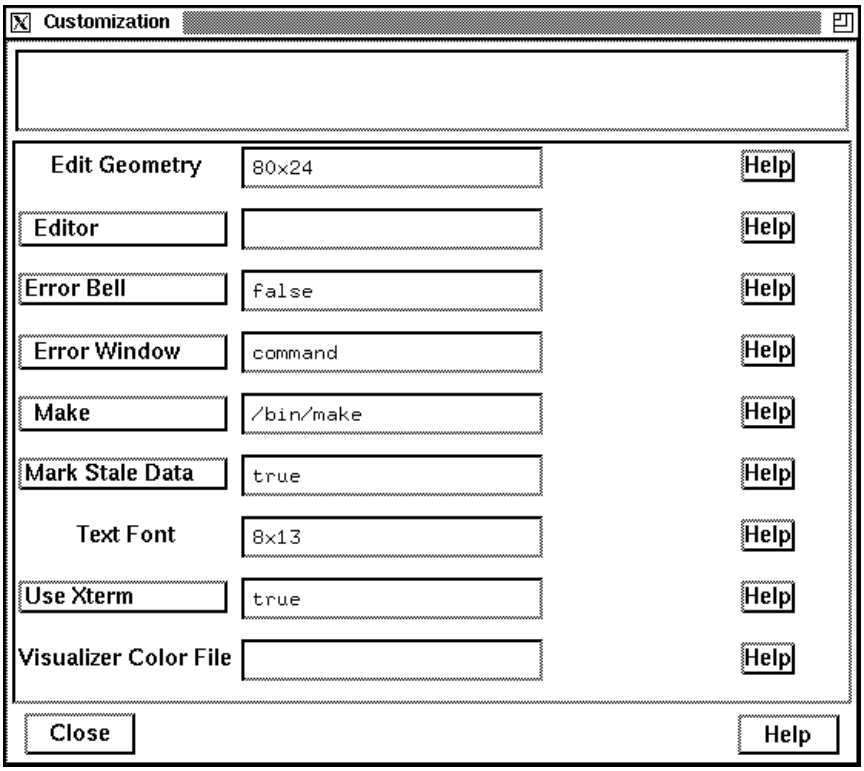

Figure 39. The Customize window.

9.3.1 How to Change a Setting

On the left of the **Customize** window are the names of the resources. Next to each resource is a text-entry box that contains the resource's setting (if any). To the right of the fields are **Help** buttons. Clicking on a **Help** button or anywhere in the text-entry field displays help about the associated resource in the box at the top of the window.

The way you set a value for a resource differs depending on the resource:

- For **Edit Geometry**, **Text Font**, and **Visualizer Color File**, you enter the setting in the resource's text-entry box.

Copyright © 1991–1994 Thinking Machines Corporation

- For **Editor**, **Error Window**, and **Make**, you can left-click on the button labeled with the resource's name. This displays a menu of choices for the resource. Clicking on one of these choices displays it in the resource's text-entry box. For **Editor** and **Make**, you can also enter the setting directly in the text-entry box.

- For **Error Bell** and **Use Xterm**, there are only two possible settings, **yes** and **no**; clicking on the button labeled with the resource's name toggles the current setting.

Whenever you make a change in a text-entry box, **Apply** and **Cancel** buttons appear to the right of it. Click on **Apply** to save the new setting; it takes effect immediately. Click on **Cancel** to cancel it; the setting changes back to its previous value.

Click on **Close** or press the **Esc** key to close the **Customize** window.

9.3.2 The Resources

Edit Geometry — Use this resource to specify the X geometry string for the editor created by the **Edit** and **Email** selections from the **Utilities** menu. The geometry string specifies the number of columns and rows, and optionally the left and right offsets from the corner of the screen. The Prism default is 80x24 (that is, 80 rows and 24 columns). See your X documentation for more information on X geometries.

Editor — Use this resource to specify the editor that Prism is to invoke when you choose the **Edit** or **Email** selection from the **Utilities** menu, or when you display a file via the **Release Notes** or **Bug Updates** selections from the **Doc** menu. Click on the **Editor** box to display a menu of possible choices. If you leave this field blank, Prism uses the setting of your **`EDITOR`** environment variable to determine which editor to use.

Error Bell — Use this resource to specify how Prism is to signal errors. Choosing **yes** tells Prism to ring the bell of your workstation. Choose **no** (the Prism default) to have Prism flash the screen instead.

Error Window — Use this resource to tell Prism where to display Prism error messages. Choose **command** (the Prism default) to display them in the command window. Choose **dedicated** to send the messages to a dedicated window; the window will be updated each time a new message is received. Choose **snapshot** to send each message to a separate window.

Version 2.0, April 1994
Copyright © 1991–1994 Thinking Machines Corporation

Make — Use this resource to tell Prism which **`make`** utility to use when you choose the **Make** selection from the **Utilities** menu. The Prism default is the standard UNIX **`make`** utility, **`/bin/make`**. Click on the **Make** box to display a menu of possible choices.

Mark Stale Data — Use this resource to tell Prism how to treat the data in a visualizer that is out-of-date (because the program has continued execution past the point at which the data was displayed). Choose **true** (the default) to have Prism draw diagonal lines over the data; choose **false** to leave the visualizer's appearance unchanged.

Text Font — Use this resource to specify the name of the X font that Prism is to use in displaying the labels of histogram bars and text in visualizers. The default, 8x13, is a 12-point fixed-width font. To list the fonts available on your system, issue the UNIX command **`xlsfonts`**. Specifying a font much larger than the default can cause display problems, because Prism doesn't resize windows and buttons to accommodate the larger font.

Use Xterm — Use this resource to tell Prism what to do with the I/O of a program. Specify **yes** (the Prism default) to tell Prism to create an Xterm in which to display the I/O. Specify **no** to send the I/O to the Xterm from which you started Prism.

Visualizer Color File — Use this resource to tell Prism the name of a file that specifies the colors to be used in colormap visualizers. If you leave this field blank, Prism uses gray for elements whose values are not in the context you specify; for elements whose values are in the context, it uses black for values below the minimum, white for values above the maximum, and a smooth spectral map from blue to red for all other values.

The file must be in ASCII format. Each line of the file must contain three integers between 0 and 255 that specify the red, green, and blue components of a color. To get a list of some of the colors available on your system, you can consult the file **`/usr/lib/X11/rgb.txt`**.

The first line of the visualizer color file contains the color that is to be displayed for values that fall below the minimum you specify in creating the visualizer. The next-to-last line contains the color for values that exceed the maximum. The last line contains the color used to display the values of elements that are not in the context specified by the user in a **`where`** statement.

Copyright © 1991–1994 Thinking Machines Corporation

Prism uses the colors in between to display the values falling between the minimum and the maximum. For example:

```
0       0       0
255     0       0
255     255     0
0       255     0
0       255     255
0       0       255
255     0       255
255     255     255
100     100     100
```

Like the default settings, this file specifies black for values below the minimum, white for values above the maximum, and gray for values outside the context. But the file reverses the default spectral map for other values: from lowest to highest, values are mapped red-yellow-green-cyan-blue-magenta.

9.3.3 Where Prism Stores Your Changes

Prism maintains a file called **`.prism_defaults`** in your home directory. In it, Prism keeps:

- changes you make to Prism via the **Customize** utility
- changes you make to the tear-off region
- changes you make to the size of the panes within the main Prism window

Do not attempt to edit this file; make all changes to it through Prism itself. If you remove this file, you get the default configuration the next time you start up Prism.

Version 2.0, April 1994
Copyright © 1991–1994 Thinking Machines Corporation

9.4 Changing Prism Defaults in Your X Resource Database

As mentioned in the previous section, you can change the settings of many Prism resources either by using the **Customize** utility or by adding them to your X resource database. This section describes how to add a Prism resource to your X resource database. An entry is of the form

resource-name **:** *value*

where *resource-name* is the name of the Prism resource, and *value* is the setting. Table 2 lists the Prism resources.

Table 2. Prism resources.

Resource	Use
Prism.cppPath	Specify the path to your C preprocessor.
Prism.dialogColor	Specify the color for dialog boxes.
Prism.editGeometry	Specify the size and placement of editor window.
Prism.editor	Specify the editor to use.
Prism.errorBell	Specify whether the error bell is to ring.
Prism.errorwin	Specify the window to use for error messages.
Prism*fontList	Specify the font for labels, menu selections, etc.
Prism.helpColor	Specify the color for help windows.
Prism.mainColor	Specify the main background color for Prism.
Prism.make	Specify the **`make`** utility to use.
Prism.markStaleData	Specify how Prism is to mark stale data in visualizers.
Prism.pathMap	Provide mapping between local and remote paths for use in remote execution**.**
Prism.spectralMapSize	Specify the size of the default spectral color map for color visualizers.
Prism.textBgColor	Specify the background color for widgets containing text.

Copyright © 1991–1994 Thinking Machines Corporation

Table 2. Prism resources *(cont'd)***.**

Resource	Use
Prism.textFont	Specify the text font to use for certain labels.
Prism.textManyFieldTranslations	Specify the keyboard translations for dialog boxes that contain several text fields.
Prism.textMasterColor	Specify the color used to highlight the master pane in a split source window.
Prism.textOneFieldTranslations	Specify the keyboard translations for dialog boxes that contain one text field.
Prism.useXterm	Specify whether to use a new Xterm for I/O.
Prism.vizColormap	Specify the colors to be used in colormap visualizers.
Prism*XmText.fontList	Specify the text font to use for most running text.

Note that the defaults mentioned in the sections below are the defaults for Prism as shipped; your system administrator can change these in Prism's file in your system's **`app-defaults`** directory.

Note also that commands-only Prism is not aware of the settings of any Prism resources, unless they are contained in Prism's **`app-defaults`** file. This matters only for the resources **Prism.cppPath** and **Prism.pathMap**.

9.4.1 Adding Prism Resources to the Resource Database

The X resource database keeps track of default settings for programs running under X. Use the **`xrdb`** program to add a Prism resource to this database. An easy way to do this is to use the **`-merge`** option and to specify

Copyright © 1991–1994 Thinking Machines Corporation

the resource and its setting from the standard input. For example, the following command specifies a default editor (the resource is described below):

```
% xrdb -merge
  Prism.editor: emacs
   Ctrl-d
```

Type **Ctrl-d** to signal that there is no more input. Note that you must include the **-merge** option; otherwise, what you type replaces the contents of your database. The new settings take effect the next time you start Prism.

Another way to add your changes is to put them in a file, then merge the file into the database. For example, if your changes are in **prism.defs**, you could issue this command:

```
% xrdb -merge prism.defs
```

Consult your X documentation for more information about **xrdb**.

9.4.2 Specifying the Editor and Its Placement

Use the **Prism.editor** resource to specify the editor that Prism is to invoke when you choose the **Edit** or **Email** selection from the **Utilities** menu (or issue the corresponding commands).

Use the resource **Prism.editGeometry** to specify the X geometry string for the editor created by the **Edit** selection from the **Utilities** menu. The geometry string specifies the number of columns and rows, and the left and right offsets from the corner of the screen.

You can also change the settings of these resources via the **Customize** utility; see Section 9.3 for more information.

9.4.3 Specifying the Window for Error Messages

Use the **Prism.errorwin** resource to specify the window to which Prism is to send error messages. Predefined values are **command**, **dedicated**, and **snapshot**. You can also specify your own name for the window.

You can also change the setting of this resource via the **Customize** utility; see Section 9.3.

Copyright © 1991–1994 Thinking Machines Corporation

9.4.4 Changing the Text Fonts

You may need to change the fonts Prism uses if, for example, its fonts aren't available on your system. Use the resources described below to do this. To list the names of the fonts available on your system, issue the UNIX **`xlsfonts`** command. You should try to substitute a font that is about the same size as the Prism default; substituting a font that is much larger can cause display problems, since Prism does not resize windows and buttons to accommodate the larger font.

Use the **Prism.textFont** resource to specify the font that Prism is to use in displaying the labels of histograms and text in visualizers. By default, Prism uses a 12-point fixed-width font for this text.

You can also change the setting of this resource via the **Customize** utility; see Section 9.3.

Use the **Prism*XmText.fontList** resource to change the font used to display most of the running text in Prism, such as the source code in the source window. By default, Prism uses a 12-point fixed-width font for this text.

Use the **Prism*fontList** resource to change the font used for everything else (for example, menu selections, pushbuttons, and list items). By default, Prism uses a 14-point Helvetica font for this text.

9.4.5 Changing Colors

Prism provides several resources for changing the default colors it uses when it is run on a color workstation. To get a list of some available colors, you can consult the file **`/usr/lib/X11/rgb.txt`**.

Changing the Colors Used for Colormap Visualizers

Use the **Prism.vizColormap** resource to specify a file that contains the colors to be used in colormap visualizers. You can also change the setting of this resource via the **Customize** utility; see Section 9.3. See Section 9.3.2 for a discussion of how to create a visualizer color file.

Use the resource **Prism.spectralMapSize** to specify how large the default spectral color map is to be for colormap visualizers. The default is 100 entries. You would typically use this resource to specify fewer entries, if this

Version 2.0, April 1994
Copyright © 1991–1994 Thinking Machines Corporation

number causes problems on your workstation. To set the default to 50, for example, set the resource in your X resource database as follows:

```
Prism.spectralMapSize: 50
```

Changing Prism's Standard Colors

Use the **Prism.helpColor** resource to change the background color of help windows.

Use the **Prism.dialogColor** resource to change the background color of dialog boxes.

Use the **Prism.textBgColor** resource to change the background color for text in buttons, dialog boxes, help windows, etc. Note that this setting overrides the setting of the X toolkit **`-bg`** option.

Use the **Prism.textMasterColor** resource to change the color used to highlight the master pane when the source window is split.

Use the **Prism.mainColor** resource to change the color used for just about everything else.

The defaults are:

```
Prism.helpColor: bisque3
Prism.dialogColor: Thistle
Prism.textBgColor: snow2
Prism.textMasterColor: black
Prism.mainColor: light sea green
```

9.4.6 Changing Keyboard Translations

You can change the keys and key combinations that Prism translates into various actions. In general, doing this requires an understanding of X and Motif programming. You may be able to make some changes, however, by reading this section and studying the defaults in Prism's file in your system's **`app-defaults`** directory.

Copyright © 1991–1994 Thinking Machines Corporation

Changing Keyboard Translations in Text Widgets

Use the **Prism.textOneFieldTranslations** resource to change the default keyboard translations for dialog boxes that contain only one text field. Its default definition is:

```
Prism.textOneFieldTranslations: \
    <Key>osfDelete: delete-previous-character() \n\
    <Key>osfBackSpace: delete-previous-character() \n\
         Ctrl<Key>u: erase_to_beginning() \n\
         Ctrl<Key>k: erase_to_end() \n\
         Ctrl<Key>d: delete_char_at_cursor_position() \n\
         ctrl<Key>f: move_cursor_to_next_char() \n\
         Ctrl<Key>h: move_cursor_to_prev_char() \n\
         Ctrl<Key>b: move_cursor_to_prev_char() \n\
         Ctrl<Key>a: move_cursor_to_beginning_of_text() \n\
         Ctrl<Key>e: move_cursor_to_end_of_text()
```

(The definitions with **osf** in them are special Motif keyboard symbols.)

Use the **Prism.textManyFieldTranslations** resource to change the default keyboard translations for dialog boxes that contain several text fields. Its default definition is:

```
Prism.textManyFieldTranslations: \
    <Key>osfDelete: delete-previous-character() \n\
    <Key>osfBackSpace: delete-previous-character() \n\
    <Key>Return: next-tab-group() \n\
    <Key>KP_Enter: next-tab-group() \n\
         Ctrl<Key>u: erase_to_beginning() \n\
         Ctrl<Key>k: erase_to_end() \n\
         Ctrl<Key>d: delete_char_at_cursor_position() \n\
         Ctrl<Key>f: move_cursor_to_next_char() \n\
         Ctrl<Key>h: move_cursor_to_prev_char() \n\
         Ctrl<Key>b: move_cursor_to_prev_char() \n\
         Ctrl<Key>a: move_cursor_to_beginning_of_text() \n\
         Ctrl<Key>e: move_cursor_to_end_of_text()
```

If you make a change to any field in one of these resources, you must copy all the definitions.

Version 2.0, April 1994
Copyright © 1991–1994 Thinking Machines Corporation

Changing General Motif Keyboard Translations

Prism uses the standard Motif translations that define the general mappings of functions to keys. They are shown below.

```
*defaultVirtualBindings: \
  osfActivate :                <Key>Return \n
  osfAddMode :                 Shift <Key>F8 \n
  osfBackSpace :               <Key>BackSpace \n\
  osfBeginLine :                Key>Home \n\
  osfClear :                   <Key>Clear \n\
  osfDelete :                  <Key>Delete \n\
  osfDown :                    <Key>Down \n\
  osfEndLine :                 <Key>End \n\
  osfCancel :                  <Key>Escape \n\
  osfHelp :                    <Key>F1 \n\
  osfInsert :                  <Key>Insert \n\
  osfLeft :                    <Key>Left \n\
  osfMenu :                    <Key>F4 \n\
  osfMenuBar :                 <Key>F10 \n\
  osfPageDown :                <Key>Next \n\
  osfPageUp :                  <Key>Prior \n\
  osfRight :                   <Key>Right \n\
  osfSelect :                  <Key>Select \n\
  osfUndo :                    <Key>Undo \n\
  osfUp  :                     <Key>Up
```

To change any of these, you must edit its entry in this resource. For example, if your keyboard doesn't have an **F10** key, you could edit the **osfMenuBar** line and substitute another function key.

Note these points in changing this resource:

- All entries in the resource must be included in your resource database if you want to change any of them; otherwise the omitted entries are undefined.
- The entries in this resource apply to all Motif-based applications. If you want your changes to apply only to Prism, change the first line of the resource to **Prism*defaultVirtualBindings**.

Copyright © 1991–1994 Thinking Machines Corporation

9.4.7 Changing the Xterm to Use for I/O

By default, Prism creates a new Xterm for input to and output from a program. Set the **Prism.useXterm** resource to **false** to tell Prism not to do this. Instead, I/O will go to the Xterm from which you invoked Prism. You can also change the setting of this resource via the **Customize** utility; see Section 9.3.

9.4.8 Changing the Way Prism Signals an Error

By default, Prism flashes the command window when there is an error. Set the resource **Prism.errorBell** to **true** to tell Prism to ring the bell of your workstation instead. You can also change the setting of this resource via the **Customize** utility; see Section 9.3.

9.4.9 Changing the make Utility to Use

By default Prism uses the standard UNIX **`make`** utility, **`/bin/make`**. Use the resource **Prism.make** to specify the pathname of another version of **`make`** to use. You can also change the setting of this resource via the **Customize** utility; see Section 9.3.

9.4.10 Changing the Way Prism Treats Stale Data in Visualizers

By default Prism prints diagonal lines over data in visualizers that has become "stale" because the program has continued execution from the spot where the data was collected. Set the resource **Prism.markStaleData** to **false** to tell Prism not to draw these diagonal lines. You can also change the setting of this resource via the **Customize** utility; see Section 9.3.

9.4.11 Specifying Mappings between Paths for Remote Execution

When using Prism's remote execution feature (currently available only in Node Prism), you may sometimes need to specify a mapping between the path that a file or directory has on the local computer (where you are running Prism) and its path on the remote computer (the partition manager on

Version 2.0, April 1994
Copyright © 1991–1994 Thinking Machines Corporation

which you are executing your program). Use the resource **Prism.pathMap** to list these mappings. For example:

```
Prism.pathMap:local_path1 remote_path1      \
              local_path2 remote_path2      \
              local_pathn remote_pathn      \
```

(The \ is a continuation character and not required syntax.)

Note that you can override these mappings, or provide additional mappings, by issuing the **pathmap** command. See Section 10.2.3.

9.5 Initializing Prism

Use the **.prisminit** file to initialize Prism when you start it up. You can put any Prism commands into this file. When Prism starts, it executes these commands, echoing them in the history region of the command window.

When starting up, Prism first looks in the current directory for a file called **.prisminit**. If the file is there, Prism uses it. If the file isn't there, Prism looks for it in your home directory. If the file isn't in either place, Prism starts up without executing a **.prisminit** file.

The **.prisminit** file is useful if there are commands that you always want to execute when starting Prism. For example:

- If you always want to log command output, put a **log** command in the file; see Section 2.7.4.
- If you want to use your own aliases for Prism commands, put the appropriate **alias** commands in the file; see Section 9.2.

Note that you don't need to put **pushbutton** or **tearoff** commands into the **.prisminit** file, because changes you make to the tear-off region are automatically saved when you leave Prism; see Section 9.1.

In the **.prisminit** file, Prism interprets lines beginning with **#** as comments. If **** is the final character on a line, Prism interprets it as a continuation character.

Copyright © 1991–1994 Thinking Machines Corporation