17

Chapter Seventeen
BUILDING PRESENTATIONS

In earlier chapters, we've covered in detail creating layout files, building animation trace files, and instrumenting simulation models to produce trace streams to drive animations. In this chapter, we'll see how to use presentation commands to combine animations, slides, and sound into a Proof presentation. Appendix C contains a reference listing of the presentation commands.

Why Should You Create a Proof Presentation?

Proof can be an invaluable tool during the design of a system. When a design is finished, or at critical stages of the design process, you may have to show animations to others, e.g., your boss, your customer, your sponsor, etc. You may have produced many minutes' worth of animations, but virtually no one will want to watch them all from start-to-finish. When you build a presentation, you must select representative "clips" of animation that best illustrate the points you want to make. If you anticipate that the boss will ask "What about the loading dock during peak delivery period?" you'd better have some examples in your presentation. If you do, it makes you look as though you know what you're doing. On the other hand, if you have to take a minute or two to locate the right files, fast forward to the appropriate time, and zoom in on the area of interest, you'll lose the boss's attention.

A well-structured presentation makes it easy for users of your animation to see what they want to see, and it makes it easier for you to make the points you want to make. The combination of slides and animations is very effective for communicating ideas.

Who Runs the Presentation?

Throughout the chapter, we'll use the term "user" to refer to whomever might operate a presentation. Often the user will be you (giving a presentation to others). With appropriate

instructions, the users might also include your customers, sponsors, managers, and others.

Presentation Basics

Proof can perform the following actions in a presentation:

- Display a slide (static image)
- Run all or part of an animation
- Play a sound file
- Wait for time (wall-clock, not simulated) to elapse or for user intervention
- Use special effects to transition between slides or animations
- Display a custom menu for navigating through a hierarchical presentation
- Invoke another program or issue an operating system command

These actions are specified as commands in a *presentation script file* (.PSF file). A presentation script file is an ASCII text file, usually fairly compact, that you create using a text editor or word processor. (Be sure to save the file as text only or ASCII.)

Linear vs. Hierarchical Presentations

Proof provides for two kinds of presentations, linear and hierarchical. Linear presentations comprise a list of commands that is executed from top to bottom. A user can jump ahead or jump back in a linear presentation, but only one step at a time. In hierarchical presentations, commands are organized as a group (tree), with optional subgroups (subtrees), allowing the user quickly navigate through a presentation, homing in on areas of interest.

Linear Presentations

If a presentation script file has no group or subgroups, it is by definition a *linear* presentation. Such a presentation is simply an ordered list of commands. Each command is executed exactly once, in sequence, unless a user intervenes.

The simplest presentation you could design would be an automatically paced display of slides and animations that assumes no user intervention. The commands you would use include

- slide (to display a slide)
- run (to show all or part of an animation)
- play (to play a sound file)
- wait (to delay for a fixed "wall clock" time interval)
- wipe (to control the transition effect leading to the next run or slide)
- end (to mark the end of the script file; required)
- exit (to exit Proof; optional)

Whether you plan for it or not, a user can control the pace of the presentation, stop and restart the presentation, and jump forward and backward through the presentation script file command list. Jumping forward and backward is covered in the description of the Presentation Mode toolbar later on in this chapter.

A linear presentation is a "forced march" through material you have selected. If you can anticipate exactly those things your users need to see and the order in which they should see them, a linear presentation may be adequate. However, it's rarely possible to anticipate exactly what people will want to see. Human beings are naturally inquisitive creatures, and everyone has different interests. Building a hierarchical presentation is only marginally more difficult than creating a linear presentation, and hierarchical presentations offer much better flexibility to their users.

We think you will prefer the results you get with hierarchical presentations.

Hierarchical Presentations

The structure of a hierarchical presentation is shown by Proof in a custom-built Windows Tree View that allows a user to quickly select presentation steps, and reveal or hide the presentation's subgroups. The Tree View is called a *navigation* tree. A sample navigation tree is shown in Figure 17-1.

Figure 17-1. A Sample Hierarchical Presentation's Navigation Tree

You can add a custom, hierarchical, menu to your presentation script file, providing a user with random access to presentation components. If you do this, the structure of the script file becomes more complex. (This structure is presented in detail in the "How to use group and Related Commands" section later in this chapter.)

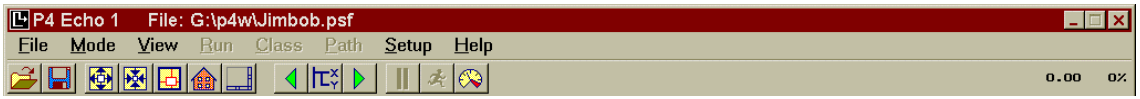
By using subgroup commands, it's possible to create a tree-like navigation structure for a presentation. In such presentations, entire subtrees can be revealed or hidden with a single mouse click. A sample navigation tree was shown in above in Figure 17-1.

Introduction to Presentation Mode

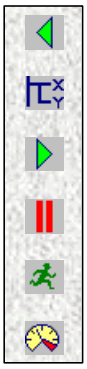
Opening a presentation script file (.PSF file) automatically places Proof into Presentation Mode. The user can open a presentation in a variety of ways. The user can choose **File, Open Presentation**; invoke Proof from a command line or launch Proof from another program, specifying a .PSF file as the file to be opened; or use the Windows Explorer to "open" a .PSF file (provided that Proof has been registered with Windows as the program to use when opening a .PSF file.)

Presentation Mode is much different from Run Mode, reflecting the differences in how the two modes are used. There are several special buttons in the Presentation Mode toolbar that can control the flow of an entire presentation.

Upon entry to Presentation Mode, a user will see the following menu and toolbar:



The toolbar includes the following buttons specific to Presentation Mode:



Goes back to the previous step in the presentation

Hides / reveals a hierarchical presentation's navigation tree

Jumps ahead to the next presentation step

Stops execution of the current animation

Resumes execution of the current animation

Speeds up / slows down an entire presentation

Overriding the Pace of a Presentation

A user can override the pace of a presentation you have constructed. Some may find that the pace you had in mind is either too fast or too slow. (It's a challenge for the presentation designer, *i.e.* you, to determine an appropriate pace. The general tendency is to use a pace that's too fast. After you've previewed a presentation a number of times, it becomes old to you, and you'll tend to want things to go faster than someone else would find comfortable in a first viewing.)

If the user clicks on the  (Pace) button, Proof will ask the user to supply a pace. A value of 1.0 causes a presentation to run at "design" speed. A value of 2.0 causes a presentation to run at twice its design speed, and a value of 0.5 causes a presentation to run at half speed.

Jumping Forward and Backward

There are three ways to jump forward or backward while viewing a presentation:

- The user can click on the  (Jump Ahead) and  (Jump Back) buttons.
- The user can press the "+" (Jump Ahead) and "-" (Jump Back) keys on the numeric keypad (assuming a fully equipped keyboard).
- If *remote mouse demo control* is enabled, the right mouse button can be used to jump

ahead, and the left mouse button can be used to jump back. To enable remote mouse demo control, choose **Setup**, **Remote Mouse Demo Control**. Note that to jump ahead or back, the mouse cursor must not be positioned on top of any toolbar button or menu item. If the mouse cursor is on top of either a button or menu item, a left mouse click will invoke the specified action.

Interpretation of Forward and backward jumps

The discussion that follows is for you, the presentation designer. The user who runs the presentation will find the interface natural.

The interpretation of jumps depends upon which of three contexts in which the jump is issued:

- In a linear presentation
- In a hierarchical presentation with the navigation tree visible
- In a hierarchical presentation with the navigation tree hidden

In a linear presentation, a jump will always jump to one of the following targets:

- A slide command (the command that displays a slide)
- A play command (the command that plays a sound file)
- A run command (the command that runs an animation)

Other presentation script file commands, such as those that show transition effects, are skipped over when a user jumps backward or forward.

In a hierarchical presentation, jumps are interpreted as follows:

If the navigation tree is visible and a jump ahead is performed…

- If the currently highlighted line in the navigation tree is an unexpanded subtree (prefixed with a "+"), the subtree will be expanded, and the first line of the subtree will be highlighted.
- If the currently highlighted line in the navigation tree is an expanded subtree (prefixed with a "-"), the first line in the subtree will be highlighted.
- If the currently highlighted line in the navigation tree is a "leaf" (not a subtree), the command or sequence of commands defined for the leaf node will be executed.

If the navigation tree is visible and a jump back is performed…

- The previous line in the navigation tree is highlighted. If the currently highlighted line is the first line of the tree, a "Start-of-Script!" message is issued.

If the navigation tree is hidden, and a jump ahead is performed,…

- If there is another command remaining in a sequence of commands defined for the active (but hidden) line in the navigation tree, it is executed. For example, frequently a single entry in a navigation tree invokes a sequence of several short animations, interspersed with time delays and/or slides. Any wait commands in the sequence, which caused real time delays, are skipped.

- If there are no further commands defined for the active line in the navigation tree, the navigation tree is displayed, with the next line in the tree highlighted. If execution has reached the end of the tree, a "No Further!" message is issued.

If the navigation tree is hidden, and a jump back is performed,…

- If there is a previous command in a sequence of commands defined for the active (but hidden) line in the navigation tree, it is executed. As with jump aheads, any wait commands in the sequence are skipped.

- If there is no previous command defined for the active line in the navigation tree, the navigation tree is displayed, with the previous line in the tree highlighted. If execution has reached the top of the tree, a "Start-of-Script!" message is issued.

If the navigation tree is visible, the <ENTER> key is treated as a jump ahead, and the arrow keys can be used to move up and down among the lines of the tree.

If the foregoing rules seem overly complex, an easy rule-of-thumb is that Proof works the way the user would expect it to work. In the world of software development, we call this the Law of Least Astonishment.

Exercise 17-1: Running a Presentation

For an example of a simple, linear presentation, click **File**, **Open Presentation**, and select FUN&SUN.PSF from the EXERCISE folder. Experiment with the various Presentation Mode controls. If you'd like to examine the actual script file, click **View**, **View Proof File as Text**, **Presentation Script File**, and select FUN&SUN.PSF.

For an example of a complex, hierarchical presentation, take a look at Wolverine's Proof Demo. Unless you elected not to install this demo when you installed Proof, the easiest way to view it is to click on the "Proof Demo" icon in the Wolverine folder on your desktop.

Using Presentation Script File Commands

Displaying Slides

A Slide is a static (non-animated) picture that is part of your presentation. Proof supports three formats for Slides: .bmp, .pcx, and .rtf. Bitmap Slides must be stored in .bmp or .pcx format.

Bitmap slides must use 256-color (8-bit color depth) format.

There are many software packages that can be used to generate .bmp files. Typically such tools can produce graphically complex, highly attractive slides, incorporating anything from photographs to bar graphs, pie charts, etc. (Normally you will use .bmp files. The .pcx file format is an older format.)

You can use Proof's built-in screen capture to generate .bmp or .pcx files. (Click **File**, **Grab Screen -> Slide**.) You can also use Windows' built in screen grabbing capability, if and only if you use a 256-color desktop.

For displaying simple textual Slides in presentations, Proof also supports .rtf (Rich Text Format) files. You can generate .rtf files by using Microsoft's WordPad, which is provided as a built-in component of all Windows operating systems. You can also use Microsoft Word or other packages for generating .rtf files, but Proof does not support fancy formatting. Most of the Slides in Wolverine's Proof Demo Disk were produced using WordPad.

Slides are read and displayed virtually instantaneously by using the slide presentation script file command. You can preview Slides by clicking **File**, **Show Slide**.

The syntax of the slide presentation script file command is

slide [**stretch**] [*transition*] *filename*

If no file extension is used in the *filename*, Proof will try extensions of ".bmp", ".rtf", and ".pcx", in that order. .bmp and .pcx files must use 256-color mode (8-bit color depth.

The stretch option is supported for .bmp files only. If a .bmp Slide is larger or smaller than Proof's client window, and the stretch option is used, the Slide will be shrunk or expanded, respectively, to exactly fit. The algorithm Proof uses for stretching is unsophisticated, so depending on the extent of any given size mismatch, the results will vary.

For .pcx slides and .bmp files without the stretch option, if the Slide image is smaller than Proof's client window, the image will be centered and surrounded by a black border. If the image is larger, the right and bottom edges will be truncated as necessary.

The following optional *transition* values can be used for bringing up a slide:

- **checkerboard** brings the slide up using enlarging squares.
- **dissolve** brings the slide up via random pixel replacement.
- **sunrise** fades the screen to black and back up to full color.
- **venetian** brings the slide up using "Venetian blinds."
- **vvenetian** brings the slide up using "vertical "Venetian blinds."

Note that if the color palette of a .bmp or .pcx file differs from the color palette currently in effect at the time the slide command is executed, sunrise is forced. This is done to avoid jarring instantaneous palette changes.

How soon a Slide disappears depends on the presentation script file commands that follow the slide command.

A slide command is almost always be followed by a wait command; otherwise, the Slide may be instantaneously superseded as the result of the command that follows. For example, if two consecutive slide commands are issued with no intervening wait, the first will be seen as a momentary flash.

Running Animation Clips

An animation clip is all or part of one animation. The presentation script file command for displaying an animation clip is run. The syntax of run is:

run [*transition*]

{ *layoutfilename* [*tracefilename*] } | *

[**view** *viewname* | "*viewname*"] [**speed** *speed*]

[[*startingtime*] *endingtime*]

(Recall from Chapter 2 that braces ({}) in a syntax description are used to denote the grouping of items. The brace characters are not part of the command you use.)

The optional transitions available for use in the run command are the same as those described for the slide command above.

Here are some examples of the run command.

run steel

run steel 0 200

run steel 200

run steel **view** Entry 100 300

run * **view** "Exit Area" 500

run dissolve steel steel2

You must specify a *layoutfilename* (or use the "*" alternative; see below). If you want to use an animation trace file that has the same base filename as the layout file, then it is not necessary to specify the *tracefilename*. Otherwise, you must also specify the *tracefilename*.

The optional "*" replaces both the *layoutfilename* and the *tracefilename* and stands for "same layout and trace files as specified in the preceding run command in the presentation script file."

The optional *viewname* can be the name of any of the views that are stored in the layout file. If the *viewname* contains one or more spaces, you must enclose it in double quotes. The default view, if you don't include the VIEW option in your run command, is the Home view (or, in the case of run *, it is the view used in the preceding run command).

The optional *speed* determines the animation speed. The default *speed* is the one stored in the layout file (or, in the case of run *, it is the speed used in the preceding run command).

If you specify an *endingtime*, the animation will run until that time is reached. The default *endingtime* is the end of the animation trace file. Once the run command is complete, execution of the presentation script file will proceed without user intervention.

If you also specify a *startingtime*, then Proof will jump to that time before beginning the animation display. The default is 0 if a *layoutfilename* is specified, or "the current time" in the case of run *.

You must specify an *endingtime* in order to specify a *startingtime*. If only one number is specified, it will be interpreted as an *endingtime*.

For presentations running under Student Proof, the overall time limit imposed by the software applies only to the "wall clock" time during which animations are running. Of all the presentation script file commands, only the run command uses up this time.

*The Benefits of "Run *"*

If you use * for the filenames in a run command and do not supply a *startingtime*, then the animation specified in the preceding run command will continue, without rewinding, from the

time at which it previously stopped. The person viewing different time slices and/or Views of a complicated animation will be spared a possibly lengthy "Time Jump in Progress" period.

If you use run * and a *startingtime* that is greater than the time at which the preceding animation stopped, then the animation will jump, without rewinding, to that time and then continue. The person viewing different time slices and/or Views of a complicated animation will see a "Time Jump in Progress" but it will be briefer thanks to run *.

Note that in the cases just described (when run * does not cause the animation to rewind), the View and Speed will be unchanged unless you specify a different View and/or Speed in your run * command.

Playing Sound Files: the Play Command

Sound files can be incorporated into a presentation by means of the play command. Sound files can be used to incorporate narration, background music, special effects, and so forth, into a presentation. The syntax of the play command is as follows:

play [**asynchronous**] *filename*

The specified *filename* must be a .WAV file. File names that contain embedded blanks must be enclosed in quotes. The asynchronous keyword indicates that the playing of *filename* is to be overlapped with ongoing execution of an animation. If this option is not used, execution of the presentation is suspended until *filename* has been played in its entirety.

The following example plays a file named tada.wav:

play asynch tada

Ongoing, asynchronous .WAV file playing can be terminated as follows:

play off

Suspending a Presentation: the wait Command

The wait command causes the presentation to delay for a specified time period or else to halt completely until a user intervenes. The syntax is

wait [*delayvalue*] [*alertstring*]

Whenever a wait command is executed, the message "PRESENTATION PAUSED" is appended to the title bar of Proof's main window. When the presentation resumes, this message is removed.

The two independent options for the wait command mean there are four ways to use wait.

The first way is with no options. If you include a command of the form

wait

the presentation will pause until a user resumes it. Use this form of the wait command at points in a presentation where you want to force user intervention to resume. We call this an untimed wait. This mode of operation is very natural for most users. It helps them feel that they have control over the presentation.

If you include a command of the form

wait *alertstring*

the *alertstring* will be displayed in a dialog box centered on the screen. The *alertstring* begins with the first nonblank character after the word wait, and ends with the last nonblank character on the line. Blanks are allowed inside the *alertstring*. If you use this form of the wait command, be sure the *alertstring* does not begin with a number, which Proof would treat as a *delayvalue*.

If you include a command of the form

wait *delayvalue*

the presentation will be suspended for an amount of time equal to *delayvalue* (in seconds), *after which the presentation will automatically continue* with the next command in the script file. Note that if a user has changed the viewing pace of the presentation, delayvalue will be scaled appropriately. We call this form of the wait command a timed wait.

A user can choose to intervene during the delay time. In fact, it often makes sense to use wait *delayvalue* with a large value (e.g. 10 or 20 seconds) instead of using an untimed wait. Waits with large delay values cause a presentation to keep moving, albeit slowly, if a user is not paying attention or uncertain of what to do next. If you underestimate the time your users will need, your use of timed delays will frustrate them.

Use timed waits judiciously.

Finally, you can use

wait *delayvalue alertstring*

…which specifies a timed wait with an *alertstring*.

The Syscall Command

syscall *syscommand*

The syscall command lets you invoke another program or execute an operating system command from within your presentation. After the operating system command is executed, your presentation resumes by executing the next line in your presentation script file.

Here are some examples:

syscall myprog.exe

syscall HPRO mymodel

In the first example, a listing of your current directory appears on screen before the presentation resumes.

In the second example, the presentation will stop, and the simulation model called *mymodel* will execute under GPSS/H Professional. Then the presentation will resume.

The Title Command

title *textstring*

The title command lets you customize the upper title bar that appears across the top of the screen during the execution of a .PSF file. This feature is supported only for commercial versions of Proof. When the presentation mode menu bar is not visible, the default text in the upper title bar is "Student Proof" (or, if you are using a commercial product, the name of that product will appear in the title bar).

The *textstring* will replace the default text.

For example

title Customer Service Survey

Here "Customer Service Survey" will appear in Proof's title bar.

The Wipe Command

wipe *color* { **dissolve** | **venetian** | **vvenetian**
curtain up | **curtain down** | **curtain updown**
checkerboard | **random** }

These are options for transitioning the display to a solid color. Wipe options are described below.

fade	performs a "fade-to-black" effect (the opposite of sunrise), regardless of the *color* specified.
dissolve	performs a "dissolve" to the specified color..
venetian	performs a "closing venetian blinds" effect and ends with the specified color.
vvenetian	performs a vertically-oriented "closing venetian blinds" effect and ends with the specified color.
curtain down	performs a "closing window shade" effect and ends with the specified color. The "shade" closes from top-to-bottom.
curtain up	performs a "closing window shade" effect and ends with the specified color. The "shade" closes from bottom-to-top.
curtain updown	performs a "closing window shade" effect and ends with the specified color. The "shade" closes from top-to-bottom and bottom-to-top, meeting in the middle of the screen.
checkerboard	sets the screen to the specified color in a checkerboard pattern of squares of increasing size.
random	chooses one of the above options at random.

Here are some examples of wipe.

wipe F3 **curtain updown**

wipe F1 **venetian**

wipe F2 **fade**

Note that in the third example, the color F2 may or may not appear. If the presentation terminates after the fade, color F2 will not be seen. Similarly, if a run or slide command is processed next, F2 will never be seen. If, however, the next action taken is to display a menu, the BACKDROP color around the menu will be F2.

Comment Lines

***** *comment*

Comment lines can be very useful in script files to show where items and other subsections begin and end. Any line that begins with an asterisk (*) is a comment line.

Exiting a Presentation: the Exit Command

The exit command exits a presentation and optionally "chains" to another presentation or exits Proof. The exit command has the following form:

exit [*filename* | **system**]

An exit command with no *filename* or system keyword terminates the current presentation, but Proof remains active. If exit system is specified, execution of Proof is terminated, and the Proof window disappears. If exit *filename* is specified, Proof "chains" to the script file specified by the *filename* and begins processing the new presentation script. If no extension is given as part of the *filename*, .PSF is assumed.

Terminating a Presentation Script: the End Command

Every presentation script must be terminated by an end command:

end [*filename*]

In a linear presentation, execution of the end command terminates the current presentation script. If a *filename* is specified, Proof "chains" to the script file specified by the *filename*. If no extension is given as part of the *filename*, .PSF is assumed.

A hierarchical presentation always includes a group…endgroup command sequence. It is impossible to jump out of the group…endgroup sequence, so in a hierarchical presentation, the end command, which is required and must follow the endgroup command, can never be executed. We'll examine the format of a hierarchical presentation in detail below.

The Structure of a Hierarchical Presentation Script

A typical hierarchical presentation has the following form:

```
title Presentation Title
[ introductory actions ]
group  Menu Title
            item  Animation 1 Name
                actions
            subgroup Subgroup 2 Name
                item Animation 2.1 Name
                    actions
                item Animation 2.2 Name
                    actions
                subgroup Subgroup 2.3 Name
                    item Animation 2.3.1 Name
                        actions
                    item Animation 2.3.2 Name
                        actions
                    …
                endsubgroup
                …
            endsubgroup
            …
            subgroup Subgroup 3 Name
            …
            endsubgroup
            item Exit Presentation
            exit system
endgroup
end
```

Note that while indentation of the commands shown above is not required, it greatly increases the readability of a presentation script. Further note that there is no escape from group…endgroup; i.e., execution of a script does not fall through to the next command when an endgroup command is executed. Execution of an endgroup command forces a presentation's navigation tree to be displayed if it has been temporarily suppressed.

Presentation Navigation Windows and Navigation Trees

Most presentation scripts include a group…endgroup structure. When a group statement is executed, a navigation window, whose title is specified in the group command, appears on the screen. Within the navigation window, a navigation tree is displayed. The tree contains Items whose names are specified by item commands. The tree may also contain Subgroups whose names are specified by subgroup commands. Subgroups displayed in the tree are preceded by a small box with a "+" or "-" in it. Clicking on a box that contains a "+" causes the items and lower-level subgroups of the given subgroup to be shown. Clicking on a box that contains a "-", which is shown for an expanded subgroup, causes the subgroup to be shrunk back to just its title.

The navigation window is hidden when animations are running or slides are being shown, but reappears automatically when needed. The navigation window can be manually hidden or shown by clicking on the navigation tree icon in the toolbar of Proof's main window.
The use of group…endgroup has the following advantages:

- The structure of a presentation is visible.
- By clicking on items in the navigation tree, a viewer can quickly move around in a presentation.
- The use of subgroups allows construction of a hierarchical presentation. By expanding subgroups, a viewer can zoom in on topics of interest. By shrinking subgroups back to their titles, a viewer can zoom out to see the higher-level structure of a presentation.

Presentation scripts without group…endgroup are allowed, but we discourage the use of such "linear" scripts, for the following reasons:

- Linear scripts offer the user no visual cues as to their structure.
- The user can step back or ahead one step at a time in linear scripts, but cannot jump to arbitrary points in the script.
- Linear scripts are inherently non-hierarchical and therefore more difficult to navigate.

Compatibility with Earlier Versions of Proof

Earlier versions of Proof did not provide for hierarchical presentations. In the current version of Proof, the item command has replaced the former case command, group has replaced menu, and endgroup has replaced endmenu. The old command names are still accepted by the current version of Proof, assuring 100% compatibility.

How to Use Group and Related Commands

When you use the group command, the presentation halts until a user selects a next step from the navigation tree that appears at the center of the screen. The syntax of the group command is as follows:

group [*alertstring*]

If an *alertstring* is specified, it is displayed as the title bar of the navigation tree's window. An *alertstring* begins with the second character following the word group, and ends with the last non-blank character on the line. If you omit an *alertstring*, Proof will use "Please Select Next Step."

A group command notifies Proof that one or more item commands will follow. At this point Proof scans the presentation script file for item commands that appear between group and endgroup. Each item has a label (which can contain blanks), and all item labels are displayed as options in a custom "list-type" menu in the middle of the screen.

item *labelstring*

The required *labelstring*, which can contain blanks, is used in the navigation tree.

Each item command (with its custom label) is followed by a group of presentation script file commands that direct Proof to perform specified Actions, such as running animations, displaying Slides, playing sound files, etc.

There is no "enditem" statement. An item's group of commands ends at the next item statement. (The last group ends at the endgroup statement.)

A group must be terminated by an endgroup command:

endgroup

The endgroup command will always be the next to last command in a script file. There is no way out of a group…endgroup sequence, (other than through an exit statement), so any statements you include between an endgroup and an end command cannot be reached.

Exercise 17-2: Building a Presentation Script File

Start a new file using your text editor or word processor. Type in the following contents.

```
slide kanban.pcx
wait 10
slide pbrush.pcx
wait 10
slide rocka50.pcx
wait 10
end
```

Save the file in the exercise folder with the name simple.psf, then run Proof and choose **File**, **Open Presentation**, and select simple.psf.

Now develop a custom menu-based version of the same presentation using the syntax below. Have one custom menu option called "View the three slides" and another called "Exit this presentation."

```
group Choose an Option
*
item View the three slides
  slide kanban.pcx
  wait
  slide pbrush.pcx
  wait
  slide rocka50.pcx
  wait
*
item Exit this presentation
  exit
*
endgroup
end
```

Tips on Giving Effective Presentations – Hardware and Software

Hardware/software tips for giving presentations are given in Chapter 19.