CHAPTER 24

Shockwave and NetLingo

In This Chapter

Authoring for Shockwave

Working with NetLingo

Exporting to Java

On December 4, 1995, the World Wide Web (WWW) changed forever. Static graphics and text were replaced with animation and kiosk-like interactivity. This new technology was called Shockwave, and the key players were no less than Netscape, the creator of the Netscape Navigator Web browser, and Macromedia, the developer of Director.

This chapter explores a class of Director movie — the Shockwave movie — that can be distributed to users across the globe over the World Wide Web. To build Shockwave movies requires some effort, but creating Shockwave movies from your Director movies is extremely easy. This chapter takes you through the Internet-related Lingo commands (NetLingo) and the methods for building efficient and effective Shockwave movies.

Director Authoring Issues and Constraints

Authoring a Director movie for use on the Web is almost the same as creating multimedia for any other delivery system — with some added burden. Most Director features and Lingo commands are available for use in Web-bound movies, but there are a few exceptions.

The differences between authoring for the Web and authoring Director movies for other kinds of distribution (CD-ROM, kiosk, and so on) include:

- Minimizing file size to reduce download times across the Internet
- Avoiding Lingo commands and Director features that are not supported by Shockwave

Some Lingo commands are not supported in Shockwave for security reasons—that is, to prevent a Shockwave movie from maliciously destroying data or inappropriately retrieving data from the end user's system.

- ✦ Facing the limitations as well as the opportunities offered by using Director movies on the Web
- ✦ Learning the new Lingo commands designed specifically for Web delivery (called NetLingo commands)

Beware of the speed bumps in Web delivery

Although cable modems are available, most Internet users today are accessing the Web using 28.8 Kbps or 56 Kbps modems. We live in an impatient era. At 28.8 Kbps, your typical modem user is receiving about 2½ to 3 kilobytes of data per second of throughput. Depending on server performance, network traffic, and modem overhead, the rate drops even lower.

The issue is speed. Will the user who accesses your "Shocked site" tolerate long download times? The most beautiful image or the most intriguing animation must be viewed to be appreciated. If the end-user bails out before the download is complete, you have wasted your time and effort. As a gauge for your development efforts, remember that it takes about 25 seconds to download a small (60K) animation at 28.8 Kbps. If you are delivering a small movie (200K to 300K), the download time can run from 1.3 to almost 2 minutes or longer.

Table 24-1 lists some typical download times, but these times are based on a best-case scenario. The amount of traffic on all systems between your server and the user's browser, as well as the type of Internet connection (modem, ISDN line, T1 line, and so on), all influence the time it takes to download a file.

Table 24-1
Approximate Times for Downloading Files over the Internet

File Size	*14.4 Kbps*	*28.8 Kbps*	*64 Kbps*	*1.5 Mbps*
30K	35 sec.	12 sec.	1 sec.	1 sec.
100K	1.4 min.	40 sec.	5 sec.	1 sec.
200K	2.8 min.	80 sec.	10 sec.	2 sec.
500K	6.9 min.	3.3 min.	25 sec.	5 sec.
1MB	13.9 min.	6.7 min.	50 sec.	10 sec.

An ISDN line delivers approximately 64 to 128 Kbps. A T1 (high-speed access line) can deliver up to 1.5 Mbps (megabytes per second).

Losing some Director features

When you author Director movies for distribution on the Web using Shockwave technology, you have access to most of Director's features. However, there are some exceptions. Shockwave was designed to distribute multimedia over a unique environment — the Internet. Some Director features are not available because Web-based documents are downloaded one at a time, and for reasons of user security. The following limitations apply when you build Shockwave movies:

- You cannot change the color depth on the end-user's monitor (using the `colorDepth` property).
- You cannot include Director commands to save the movie to disk (`saveMovie`) and print the visible sprites on the Stage (`printFrom`).
- Custom menus are not available.
- The `open window` and `close window` Lingo commands are not available, so you cannot include movies in a window (MIAWs).
- Most system-related Director commands are disabled, including terminating or starting external applications (`quit` and `open`), turning off or restarting the user's computer system (`restart` and `shutdown`), and locating files or setting search paths for files on the end-user's system (`getNthFileNameInFolder`, `searchCurrentFolder`, and `searchPath`). These restrictions prevent anyone from creating a Shockwave movie that unexpectedly tampers with the user's system or the files stored on it.
- Director commands that control resource files (`openResFile` and `closeResFile`) located outside a Director movie are disabled.
- The `pasteFromClipboard` command that can enter media from the Clipboard is disabled.
- The `MCI` command that sends specific control strings to the Windows media controller is disabled.
- Shockwave movies cannot access the Tempo channel settings, such as the `Wait for...` options, or looping playback set by the Control Panel. (You can overcome this latter restriction by using Lingo scripts to cause a pause or loop within your movie.)
- Linked media referenced by a Shockwave movie must be available in either the Shockwave support folder or the Director Shockwave Media folder, and have an `http` address, or they cannot be accessed by the movie. This restriction is directly related to safety concerns for the end-user's system. Restricting access to the support and media folders protects users from a maliciously designed Shockwave movie that might cause serious system damage.

Limitations and opportunities of Web-delivered movies

Consider what you can do in a regular Director movie: You can include text, graphic images, sound, animation, and digital video. You can enable the user to interact with your movie by including editable text fields, hot spots, rollovers, and navigational buttons. All this is also true of Web-delivered movies. You can include each of these features in your Shockwave movies. You can even include multiple movies in a single HTML document and access information from several Web sites as part of a Director movie.

The good news is that almost everything that you have learned to do in Director you can use in Shockwave movies. The bad news is that, with Web movies, you're programming for a more limited environment, where access time is crucial, and where you can never be sure of the capabilities of your client's playback system. Keep these guidelines in mind as you design your movies targeted for Web delivery:

- Compressed Director movies should generally be kept to 50K to 200K. For the sake of those who download your movie, use several small movies that call other movies, rather than use a single large Shockwave movie.

Although the `open` command cannot be used to call other movies, the NetLingo command `goToNetMovie()` can call and display another Shockwave movie. The command `getNetText()` can display text from another Web document or file.

- Prior to version 6.01, Macromedia suggested limiting Shockwave movies to about three per page. Now the limit has been raised to 12 movies on a single Web page, but we recommend that you exercise some restraint. In addition to the RAM required to play your movie (based on its file size), an additional 50K to 100K of RAM is required for the Shockwave plug-in. Users with limited RAM may experience difficulties when trying to play back a page with several embedded movies. Because the Shockwave plug-in frees RAM when another page is loaded, it may be prudent to place movies on a sequence of separate Web pages rather than group them on a single HTML document.
- Always be aware that the load time for a Web document is based on all components of the page, which includes graphic images, length of the text portion of the document, and any Director movies it contains. Large graphic images coupled with Shockwave movies in a single document can greatly tax your user's patience.
- If you include more than one movie with audio on a single page, the browser may have problems sorting out the sound tracks. We suggest that you include only a single movie with sound on each Web document. As an alternative, consider having sounds that are activated with a mouse click rather than being played automatically as the movie starts.
- Beware of programming movies with long repeat loops. They tie up the end-user's processor. Try looping on the frame instead; each time you loop on the frame, you give the CPU and browser some processing time.

Working with NetLingo

The Lingo Network Extensions are commands and functions specifically designed to extend the capabilities of Director movies to encompass network (Internet and intranet) operations. The most obvious use for NetLingo commands and functions is in Shockwave movies. NetLingo, however, can also create hybrid Director movies to be distributed in traditional ways (on disk and CD-ROM) and can interact with network media (such as URLs, Shockwave movies, and network files).

NetLingo commands and functions fall into five general categories:

- ✦ Commands that start network operations
- ✦ Functions that evaluate the status of network operations
- ✦ Functions that retrieve the results of network operations
- ✦ Commands that cancel network operations
- ✦ Commands and functions that interact with the browser

Commands that start network operations

Five NetLingo commands (see Table 24-2) can start Internet-related operations: starting a Shockwave movie; retrieving text from an `http` server; preloading an `http` item; downloading a file; and jumping to a specific Web page. The `netAddress` parameter for each command identifies the path to the movie. Only the `file`, `ftp`, and `http` protocols are acceptable.

Table 24-2
Common NetLingo Commands

NetLingo Command	*Function*
`DownLoadNetThing` *`(netAddress)`*	Downloads a file from the network to a file on the local workstation. Operates in the background; while the download occurs, the current movie continues playing. After being installed on the local user's workstation, the file can be loaded into memory without the normal delays on network traffic.
`getNetText`	Retrieves a file that is read by Lingo as text. You can get the server's response with the `netTextResult`.
`postNetText` *`(netAddress, postData)`*	This command is the same as `getNetText()` with some additional parameters. `netAddress` is an `http` address; `postData` is a string or a list. When a property list is used, the information is posted in the same manner in which an HTML form is posted.

Continued

Table 24-2 *(continued)*

NetLingo Command	*Function*
`goToNetMovie (netAddress)`	Downloads and runs a Shockwave movie from the network. The new movie occupies the same Stage area as the current movie.
`goToNetPage (netAddress)`	Opens a URL on the network. Equivalent to including an `<a href=http:// netAddress > </a>` tag in an HTML document. It can display a Shockwave movie, an HTML document, or any other MIME type supported by the browser. The URL must be listed within quotation marks.
`PreloadNetThing (netAddress)`	Preloads a file into the browser's local disk cache so that the resource can run later without the normal download delay. Operates in the background; while the preload occurs, the current movie continues playing.

Going to a Net movie

When you use the `goToNetMovie` command, the current movie continues to run until the new movie is available — at which time the current movie is terminated without warning.

After issuing the command, be sure to keep the playback head moving. Your movie must be playing in order to make the file transfer.

In the following examples, the `goToNetMovie` command appears within an `on mouseUp` handler that is attached to a button. The first example includes a fully qualified URL that designates the path to the Shocked movie. In the second example, the movie is located in the same folder as the calling Web page, so only the movie's filename is required:

```
on mouseUp
  goToNetMovie "http://www.myserver.com/myMovie.dcr"
end

on mouseUp
   -- assumes file is in same directory
  goToNetMovie "myMovie.dcr"
end
```

You can also use a marker or label in the `netAddress` parameter, as shown in the following example. The marker or label is defined within the target HTML document by using the `<a name="labelName"></a>` tag:

```
on mouseUp
  goToNetMovie http://www.myserver.com/myMovie.dcr#intro
end
```

Going to a Net page

When you use the goToNetPage() command, the current movie continues to run until the new page is available — at which time the current movie is terminated without warning.

In the following two examples, the goToNetPage() command appears within an on mouseUp handler that is attached to a button. The first example includes a fully qualified URL that designates the path to a Web page. When the user clicks the button, the browser immediately jumps to the specified URL:

```
on mouseUp
  goToNetPage "http://www.myserver.com/~MyDocs/newText.html"
end
```

In the second example, the Web page is located in the same folder (on the server) as the calling Shockwave movie page, so only the HTML document's filename is required:

```
on mouseUp
  goToNetPage "newText.html"
end
```

The goToNetPage command also has an optional target parameter, as shown in this example:

```
on mouseUp
  goToNetPage "targetPage.html", "topFrame"
end
```

The target parameter ("topFrame" in the preceding example) can be the frame or window name in which you want the page loaded. If the target frame or window does not exist, it is created and filled with the document specified.

The preload "thing"

Preloading a target URL enables your movie to read the item from the disk cache, which can make the move far more responsive to the user and eliminate a potentially long download time. You can use the preLoadNetThing() command to preload a movie in the background while another movie continues to play. Used effectively, this command can greatly reduce normal network latency.

The netDone() command, discussed in the next section, determines whether the preload is complete. Be sure to include a short delay after issuing the preLoadNetThing() command and before issuing the netDone() command to give the preload a chance to start.

The preLoadNetThing() command loads only one http item at a time. To preload a page that has several elements (graphics and movies, for example), you must issue the preLoadNetThing() command once for each element to be downloaded.

Embedded objects, such as Director movies and GIF images on the target Web page, are not automatically downloaded when the `preLoadNetThing()` command is issued to display the Web page.

In theory, the user immediately sees a preloaded item because it is already loaded from the Internet and can be accessed from a local disk cache. The catch is that you cannot determine whether the local disk cache has been purged between the preload and the playback.

Two factors determine whether a preloaded item is available in the browser's disk cache: the browser's cache setting (the more kilobytes dedicated to the cache, the better the chance that the item will persist in cache) and the user's activity.

Getting text

You can use `getNetText()` to retrieve text off the Web. Specify the address of the file and then, when the operation is finished, use `netTextResult()` to get the text. Because `getNetText()` is an asynchronous function, Director continues to move on after you've called the function. It immediately returns an ID that you can use to test whether the function has completed.

The example in Figure 24-1 illustrates the use of `getNetText()`, `netError()`, `netDone()`, and `netTextResult()`. Error checking is very important when doing Web-based operations. Many things can happen, including the file could have been moved or renamed, the server could be down, the connection could time out, or the transfer could be interrupted.

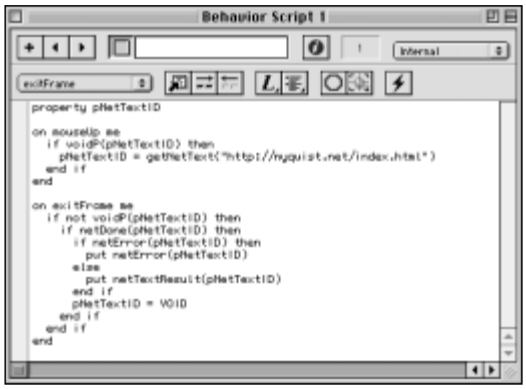

Figure 24-1: A behavior that uses four NetLingo functions

Posting text

Director has the capability to post text with the `postNetText()` function. Where `getNetText()` has a string length limit of 1 to 4 kilobytes, `postNetText()` can take arbitrarily long strings. `postNetText()` has two optional parameters besides the URL and data, as shown earlier in Table 24-2.

```
postNetText(netAddress, postData, linefeed, charset)
```

`netAddress` is the URL. `postData` is the string or property list you are posting. `linefeed` can be "Unix", "Mac", or "Win" ("Unix" is the default); any RETURN characters in the `postData` will be sent using the type for the specified operating system. `charset` is ignored unless the user is on a Japanese system; possible settings are "JIS", "EUC", "ASCII", and "AUTO".

If you try to post to a domain other than the one the movie is on, a security alert will be triggered (in Shockwave movies, not projectors).

Functions that evaluate the status of network operations

Network operations can include accessing a file, a Shockwave movie, or a URL on the Internet or an intranet. The functions listed in Table 24-3 assist you in evaluating the progress made toward the completion of a network (asynchronous) operation. (Asynchronous operations return immediately — they do not block other processing from continuing.)

Table 24-3
NetLingo Functions that Report on Network Operations

NetLingo Function	*Results*
`netDone()`	Determines if a network operation (started by `downLoadNetThing`, `getNetText`, `goToNetMovie`, `goToNetPage`, or `preLoad NetThing`) has completed. Returns true (the default) if a network operation has successfully completed or if the operation was terminated by browser error. Returns false if the operation is incomplete.
`netError()`	Returns a null or empty string until the specified network operation is complete. Upon completion, `NetError` returns an "OK" string if the operation is successful. If the operation is unsuccessful, the function returns an error number.

In the following handler, when the download is complete — that is, `netDone() = TRUE` — the `netError()` function checks for an error (not "OK"). If an error occurs, an alert box appears, listing the error:

```
If netDone() = TRUE then
  if netError() <> "OK"
    alert "Network Error:" && netError()
  end if
end if
```

Functions that retrieve the results of network operations

The three functions listed in Table 24-4 return results only after an asynchronous operation has completed. In each case, the function is available only after `netDone()` or `netError()` reports a complete operation and prior to the start of the next operation. To conserve memory, the Shockwave plug-in discards the results after the next asynchronous operation begins.

A Shockwave movie can include more than one active operation at one time. Under earlier versions of Shockwave, NetLingo used the `getLatestNetID()` function to return a unique ID for the last asynchronous operation begun. Because Net operations now return Net IDs, the `getLatestNetID()` function is no longer needed.

Table 24-4
NetLingo Functions that Return Results of Network Operations

NetLingo Function	*Results*
`netTextResult()`	Returns the text result of the last network operation. If you use the `getNetText` command, `netTextResult()` returns the text of the `http` item accessed.
`netMIME()`	Returns the MIME type of the specified `http` item.
`netLastModDate()`	Returns a text string that indicates the last date a downloaded item (using the `getNetText` or `preLoadNetThing` command) was modified based on the item's `http` header.

Canceling a partially complete network operation

NetLingo includes a command that can cancel a network operation that is in progress without waiting for completion or a result. To identify the operation to cancel, the `netAbort()` command can use the net ID for the operation or can reference the target URL of the operation. Using the net ID is always preferable. In the following handler, a Shockwave movie is preloaded, and the net ID for the operation is set to `currentNetID`:

```
on startMovie
  set currentNetID =
preloadNetThing("http://www.myserver.com/myMovie.dcr")
end
```

Because the operation's net ID has been set to `currentNetID`, you can attach the following handler to a button that, when clicked, will cancel the network operation:

```
on mouseUp
  netAbort(currentNetID)
end
```

The other form of the `netAbort` command would use a URL and be similar to the following:

```
netAbort("http://www.myserver.com/myMovie.dcr")
```

The URL must be exactly the same as the one used to initiate the `preLoadNetThing()` operation.

Commands and functions that interact with the browser

The latest version of Shockwave provides two commands and one function that interact with the end-user's Web browser. These items are listed in Table 24-5.

Table 24-5
NetLingo Commands and Functions that Interact with a Browser

Syntax	*Function or Command*	*Function*
`NetStatus`	Command	Displays a message in the status bar `message` area of the user's Web browser.
`setPref fileName, fileName`	Command	Writes a text string (`textString`) to a file (`fileName`) on the user's local hard drive. This command can only write the file to a PREFS folder (which it creates) within the user's Plug-In Support folder. The `fileName` must be a valid filename, so avoid using more than eight alpha-numeric characters. Although Macintosh and Windows 95 users can use longer filenames that include spaces and special characters, Windows 3.*x* users cannot. You should constrain filenames to the most limited environment of potential users.
`GetPref (fileName)`	Function	Returns the contents of the specified file (`fileName`) in the PREFS folder within the user's Plug-In Support folder. If the file does not exist or is empty, `getPref` returns a value of Void. The `fileName` parameter must be a valid filename on the user's operating system. (See cautions for the `setPref` command, earlier in this table.)

Cluttering up the status bar

The `netStatus()` command does not work with all browsers, but when it's available, it can be used to provide prompts to the user. In the following example, the status bar displays the message `Please click a button.` The message is displayed, in this case, as soon as the Shockwave movie loads and begins playing:

```
on startMovie
  netStatus "Please click a button."
end
```

The `netStatus()` command does not work with the Shockwave ActiveX Control for Microsoft Internet Explorer 3.0.

Leaving Shockwave cookies

The `setPref` command enables you to leave a telltale file (called a *cookie*) on the user's system. A cookie simply writes a small file on your local hard drive when you visit a Web site. The next time you visit the site the Web server can retrieve the cookie to verify that you've visited the site before. Using the following handler, the `setPref` command leaves a file named Visited on the user's local drive. The file includes a text string based on today's date (using the long date function):

```
on startMovie
  set todaysDate to the long date
  setPref "Visited", todaysDate
end
```

In the handler, note that the filename ("Visited") must be within double quotation marks. The Visited file is a text file, and it will be placed within the PREFS folder that is created by the `setPref` command within the browser's plug-in folder.

Reading Shockwave cookies

In the following handler, the text string stored in the Visited file is retrieved (returned by the `getPref()` function) and stored in the variable `lastVisit`:

```
on startMovie
  set lastVisit to getPref("Visited")
end
```

If the file Visited does not exist, the `getPref` function returns a value of Void.

The next exercise uses the movie called netlingo.dir, which can be found in the EXERCISE:CH24 (EXERCISE\CH24) folder on the CD-ROM.

In the next exercise, you work with a partial Director movie called netlingo.dir. You add scripts to the buttons (to jump to different Web addresses), attach a script that leaves behind a small cookie (a file with data), and then Shock the movie.

In its current form, the netlingo.dir movie has three buttons on the Stage. There are three states for the button: up, down, and roll.

Cast member 4 is a field cast member with a single space character (the font size is set to 9 point). You will add two new scripts in cast member slots 6 and 7. In slot 6, the script displays (to the right of the buttons) an identifying label as the mouse pointer rolls over each button. If the mouse pointer is not over a button, the label displayed is a single space character. In cast member slot 7, you use the `netStatus` command to display a message on the user's status bar and to save a file ("Visited") in the user's PREFS folder. This information can be retrieved later by using the `getPrefs` function.

Finally, you'll attach a cast member script to each of the three buttons. The script uses the `goToNetPage` command to jump to a specific URL, for Macromedia, Netscape, and Microsoft, respectively. You can modify these URLs to jump to any Internet address.

Jumping to Another Internet Address and Leaving Something Behind

1. Open the netlingo.dir movie on the CD-ROM.
2. Create the behavior shown in Figure 24-2.

   When this behavior is applied to a sprite, it asks you to input a URL and a name for that URL, as shown in Figure 24-3. When the sprite receives a `mouseUp`, it goes to that URL.

Figure 24-2: A button behavior that initiates a gotoNetPage command

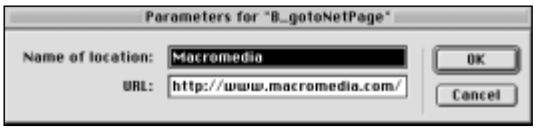

Figure 24-3: The Parameters dialog box

3. Apply the new behavior to sprites 1 through 3 (the buttons). Enter a name and URL for each of the three buttons.
4. Create a new movie script, as shown in Figure 24-4. This movie script sets a cookie with the current date. The name of the file on your disk will be "Visited.txt"; the `setPref` command appends ".txt" to the name automatically.

Figure 24-4: Getting and setting a Shockwave cookie, and putting information in the status field

5. Save the movie as **netlingo1.dir** by choosing File ➪ Save and Compact.
6. Save the movie as a Shockwave movie by choosing File ➪ Save As Shockwave Movie. Click Generate HTML and Preview in Browser.
7. Click one of the buttons, and your browser jumps to the designated Web page (only if you are online).

Creating the Ideal Shockwave Movie

Creating a small, efficiently designed movie is always an important goal. When the aim is to distribute a movie over the Web, squeezing the maximum impact out of the smallest byte-size movie is even more important. The following guidelines can help you achieve this goal:

- ✦ Try to keep the size of all cast members as small as possible — in turn, this minimizes the movie's size. You can use the Cast Info button on the Cast window to check the size of each cast member.
- ✦ To see the combined size of several cast members, press Shift+click to select them, and then click the Cast Info button.

This technique can be helpful when you are trying to break a single movie into multiple movies for faster Web distribution. It can also be used to help you locate the fat in the current movie, and which cast members might be modified to reduce the overall file size.

- One-bit images are very small and very fast. You can get more for your money from them by coloring them after they are sprites.
- Because vector shapes are defined mathematically, they are very small. Use vector shapes instead of bitmaps wherever possible. Flash is another option; because Flash movies are made up of vectors, they, too, are very small.
- Import or create small images, and then resize them on the Stage to make them appear larger. There will be degradation in your image — but, depending on how you use the graphic, the loss of quality might be acceptable. This technique can be very useful with media such as digital video and QuickTime VRs, which tend to have much larger file sizes. Having the original at half size can save you four times the file size. Remember that if you use a bitmap cast member more than once in a Director movie, it is best to import it once at its largest size and then reduce it. This method enables you to maintain the image's quality.
- Integer scaling requires fewer, faster calculations than custom scaling. What that means is that you want to use the Sprite Info dialog box to scale an object on the Stage, rather than dragging the sprite's bounding handles on the Stage.
- When you import images, choose the Remap option rather than the Dither option. Flat colors take less memory to display, and those objects will perform (animate or redraw) faster than dithered images.
- Keep the number of colors in your image as low as possible while still maintaining the image quality you desire. The number of colors used in an image is more important than color depth when compressing a movie. Don't transform images to less than 8-bit images — the size savings is in restricting the number of colors, not the bit depth.
- Another method of controlling your Director movie's size is to build backgrounds for the Stage by using small images that you can tile. Tiles can be square or rectangular. The best size tile for creating backgrounds is 16×32 pixels, or 32 pixels square. Tiles enable you to create large areas of custom texture at a very small cost in memory by using and repeating a single small cast member.
- Remember that objects created by the Tool Palette are represented by their mathematical description. Bitmap images must be stored pixel by pixel. Whenever possible, use vector objects (buttons, geometric shapes, text, and tiles) that are created by the Tool Palette, rather than bitmap images that are created in the Paint window. Vector objects use very little memory or hard disk space as compared to bitmap images.

- To change the color of individual cast members, use the Color Palette in the Tools window rather than create new palettes as cast members.
- Use text created with the Tool Palette (vector-based) rather than bitmap text (created in the Paint window) to save disk space and memory.
- Use only the characters you need when embedding a font.
- If possible, use only fonts that are on all machines. For Windows: New Courier, Arial, and Times New Roman; for Macintosh: Courier, Helvetica, and Times.
- If you want your movie to play well on the Web, test it at least on the two most common platforms: Windows 95 and Macintosh. Potentially, your Web pages will be viewed by users on both the Macintosh and the Windows platforms (not to mention Windows 98 and Windows NT).
- Sound can dramatically increase the size of a movie, but it also adds impact and flavor to a project. Sound is often necessary to convey either content or mood. It is not unusual to allocate up to 20 percent of a movie's total bytes for sound. Try to use the lowest sampling rate possible while still maintaining the quality of the sound. For speech, the lowest recommended sampling rate is 11.025kHz. For quality music playback, you will probably need to use the 22.050kHz sampling rate.
- Consider using linked Shockwave Audio (SWA) files rather then embedding large sound files in your Shockwave movie. SWA can stream across the network and begin playing as soon as a sufficient portion of the file has arrived at the user's computer.
- When you need a long, background sound track, trying looping a shorter sound. You can check the byte size of a sound loop by using the Cast window Info dialog box.
- Repeated sounds, such as found in a sound loop, can become annoying. If you use a sound loop, have it fade out or stop after playing a few times.
- Never make a visitor to your Web pages wait for a download without some indication that something is happening. Use screen text or a small Shockwave movie to alert the user that a movie is being downloaded. A small Shockwave movie with looping animation (using the `GoToNetMovie` command) is often a nice cover for the background download of a larger movie. In fact, the pre-movie can be as simple as a single `on startMovie` handler that sets the color of the Stage and then preloads or loads the second movie.
- Remember that time is relative. Folks who are accessing your movie are probably more willing to wait for a game than for a banner or fancy bullet, no matter how intriguing the movie.
- Use `preLoadNetThing` whenever possible to bring the next media object into the disk cache. When properly timed, background preloading can greatly enhance the responsiveness of your application.

- ✦ Test your Shockwave movies initially from a single folder on your Desktop before uploading them to your Web server. Open your pages from within your browser to be sure that all relative URLs work as desired. Make the necessary changes while the files are still local. If you're not careful about testing on your local drive, you can waste a lot of time transferring files that must then be fixed and transferred again. If your NetLingo doesn't work on the local drive, it won't work on the server.
- ✦ Unless you want the movie's soundtrack to go on forever (or until the user shuts down the browser), be sure to stop all sounds in the `stopMovie` handler. The same caution applies when using `goToNetMovie` to jump to a second Shockwave movie. Be sure to shut down the sound.
- ✦ If using `preLoadNetThing` and `goToNetMovie` causes the movie to be loaded twice, try calling the movie in the `goToNetMovie` instruction, using just the filename without the path.

Exporting to Java

Director can (with constraints) be exported to Java. There might be times when you do not want the user to download the Shockwave plug-in. Depending on the characteristics of your movie, Java may be an option. Both Netscape's and Microsoft's browsers support Java. The exported Java should run in any browser that supports Java 1.0.2.

The Java Export is good for advertising banners, animation, and simple games. It is best to start with the idea that you are creating a movie to be played in Java, as opposed to taking an existing one and converting it.

Java is a general-purpose, platform-independent programming language. It is not a specialized, multimedia-authoring tool, such as Director. Animation made in Java will run slower than in Shockwave because the Shockwave plug-in is written and optimized for each platform.

While developing the movie, you should test it often in your target browsers. If you know how to program in Java, you can export source code. This enables you to add more Java-specific code. Another benefit is that Java plays on 68K Macs, whereas Director no longer supports that processor.

What is a Java applet?

After writing Java code, you compile it. You can compile it to run as an application or as an applet to run in a browser, similar to the way you can make a projector with Director or create a Shockwave movie. Just as Shockwave movies have limitations due to security, so does an applet. The advantage with Java is that it is (supposed to be) platform-independent. Although it's not quite there yet, it is much easier to develop on multiple platforms in Java than in the C++ programming language.

Limitations

The Java Export does not support every Director feature. In fact, there are more than 400 commands not supported by the Java Export Xtra. When you run a check on an existing Director movie, the error list can be daunting (see Figure 24-5).

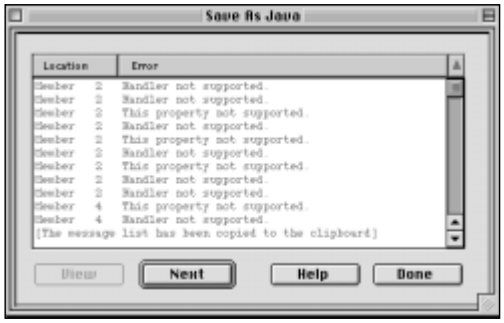

Figure 24-5: You might be surprised at the number of errors you get when checking an existing movie.

This is one reason it is easier to develop from the beginning with the knowledge that you will be exporting to Java. Although all these unsupported Lingo commands might seem like a huge downside for Lingo programmers, it won't affect you if you use Director for animation or simple interactivity. There is no better animation tool for Java than Director.

Other limitations to consider are that Java Beans must be rewritten as a class that inherits from member and sprite classes, and AWT is out because it draws differently than Director's custom Java engine.

Java options

The Java Export Xtra has many options. The first two you confront are Source or Compiled Java (see Figure 24-6).

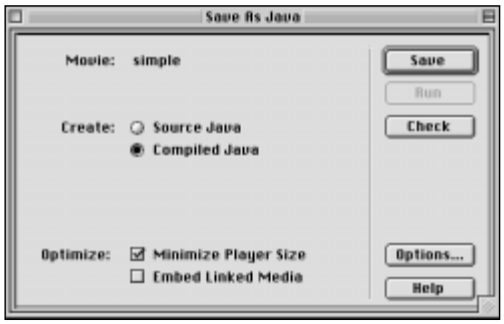

Figure 24-6: The Save As Java dialog box

Source code is in the form of text files with the .JAVA extension. You can open source code files and edit them if you are familiar with Java. Figure 24-7 shows source code from a simple movie. This can be done in a simple text editor or in a development environment made for Java programming. Editing Source and working with the API is a topic that could fill an entire book. The documentation comes with some examples of subclassing the sprite and member classes. Compiled Java is in the form of byte code, so it cannot be edited in this state.

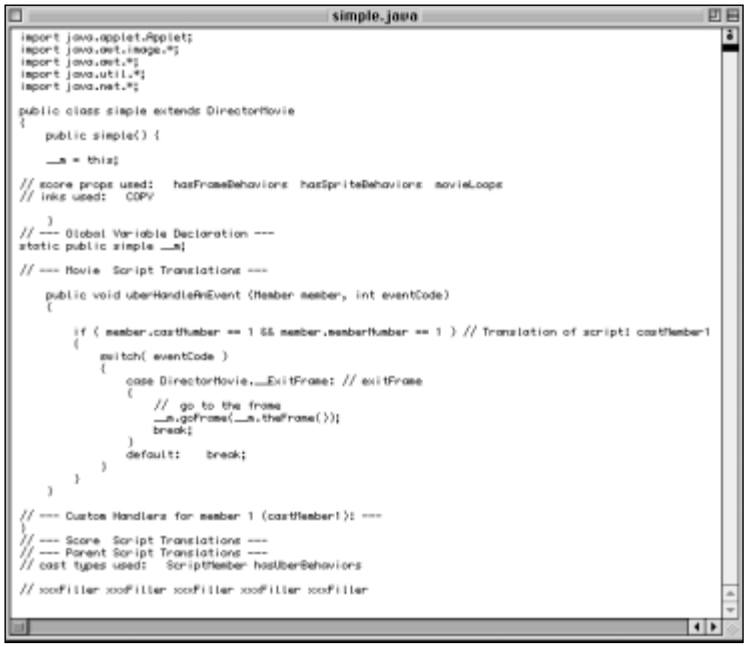

Figure 24-7: A snippet of the Java source code produced by a very simple Director movie

Minimizing the player size removes features from the player that the movie does not use. The downside of this is that removing a feature may cause errors from the Xtra or the compiler, or it may affect playback.

The next option is whether you want to embed linked files. Embedded media are placed in the DJR media file (it is given the same name as the movie file, but with the extension .DJR). Embedding media improves the download speed by reducing the number of connections that need to be made to the server. On the other hand, leaving the media as linked enables you to update the media without having to re-export the movie.

Clicking the Option button presents the dialog box shown in Figure 24-8.

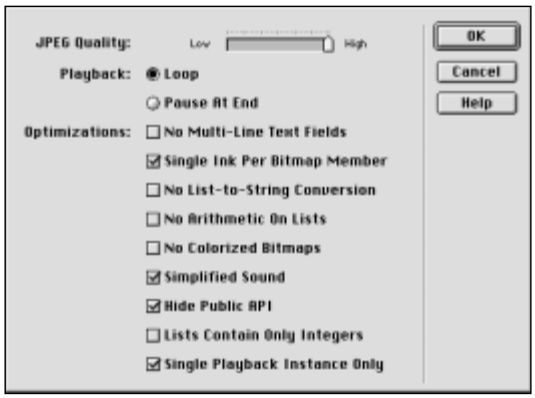

Figure 24-8: Options to optimize Java size and performance

Let's explore the options shown in Figure 24-8 one by one:

- **No Multi-Line Text Fields:** Only single line fields are supported when this option is selected.
- **Single Ink Per Bitmap Member:** This removes code from the player, which improves performance of bitmap members that are used with different inks.
- **No List-to-String Conversion:** With this on, you cannot perform the statement `string(myList)` where `myList` is a variable holding a list or a list literal.
- **No Arithmetic On Lists:** Removes code that handles operations such as `[3, 2] * 2`.
- **No Colorized Bitmaps:** When enabled, the forecolor and backcolor of bitmaps are ignored.
- **Simplified Sound:** Removes code that enables an applet to play multiple occurrences of the same sound continuously.
- **Hide Public API:** Do not select this option if you are adding your own Java code; otherwise, you can save space by choosing this option.
- **Lists Contain Only Integers:** If you are using only integers in your lists, you can select this option to increase performance. You can't do a `repeat with... in` if this option is selected.
- **Single Playback Instance Only:** Multiple applets can use the same player; if you select this option, the code that supports multiple applets is removed.

Placing Java in Lingo

Obviously, a strong knowledge of Java is important when you are developing movies that you intend to export to Java source and then modify the source. Aside from modifying source files, it is possible to insert Java code in Lingo scripts. This code does not execute in Director, but in the Java applet:

```
-- java begin
   -- System.err.println("Hello from Java!")
-- java end
```

Java is a typed language; when Lingo is exported to Java, the type of variable is guessed. If the type cannot be determined, the `LVal` type is used. Suppose that you have the following Lingo code:

```
set i to 3
```

The variable *x* will be declared an integer in the Java code—it appears in Java source as follows:

```
int i;
i = 3;
```

When you use variables that contain parent scripts, it is more efficient to explicitly declare the type. If you have a parent script called game, you declare the variable `myGame` in Lingo as follows:

```
-- java type game myGame
set myGame to new(script "game")
```

You can also put Lingo-only code in your scripts. This code will only execute in a Director movie, but it will not export to Java:

```
-- Lingo begin
   put "Hello from Lingo!"
-- Lingo end
```

Summary

Some of the things you learned in this chapter include the following:

- ✦ Shockwave technology is composed of two distinct elements: the compression engine and the Shockwave plug-in and ActiveX control.
- ✦ You can incorporate almost all Director features and commands into a Shockwave movie.

- Size is the most important constraint when creating a Shockwave movie. Because large movies take longer to download over the Internet or over an intranet, keeping files small is a major factor in creating responsive, efficient movies.
- Due to security concerns and the unique nature of network operations, some Director features are not available in Shockwave movies. For instance, access to the `saveMovie`, `printFrom`, `quit`, `restart`, `open`, `shutdown`, `pasteFromClipboard`, `open window`, and `close window` commands is prohibited.
- NetLingo commands are available to download (`downLoadNetThing`) or preload (`preLoadNetThing`) Shockwave movies or other network files, to retrieve text (`getNetText`) from a network file, and to play a Shockwave movie (`goToNetMovie`) or jump to a network address (`goToNetPage`).
- NetLingo includes two functions, `NetDone()` and `NetError()`, that determine the status of a network operation.
- The NetLingo `NetAbort` command interrupts a partially complete network operation.
- Three NetLingo functions are available to return the results of a network operation. The most commonly used NetLingo function is `netTextResult`, which returns the result of a `getNetText` operation.
- The final three NetLingo commands communicate with the browser. The `netStatus` command displays text on the status line, while the `setPref` command and the `getPref` function write and retrieve information, respectively, to a file on the user's local drive.
- Director movies can be exported to Java source or byte code.
- The version of Java supported is 1.0.2.
- Many Director commands are not supported in Java, so it is better to start from scratch when building a movie to be exported as Java.

In the next, chapter we delve into using Xtras in Director, specifically, the FileIO Xtra.

✦ ✦ ✦