Director in a Nutshell

by Bruce A. Epstein

Copyright © 1999 Bruce A. Epstein. All rights reserved.
Printed in the United States of America.

Published by O'Reilly & Associates, Inc., 101 Morris Street, Sebastopol, CA 95472.

Editor: Tim O'Reilly

Production Editor: Nancy Wolfe Kotary

Printing History:

March 1999: First Edition.

Nutshell Handbook, the Nutshell Handbook logo, and the O'Reilly logo are registered trademarks of O'Reilly & Associates, Inc. The association of the image of an ostrich and the topic of Director is a trademark of O'Reilly & Associates.

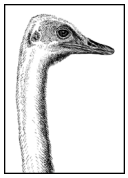
Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and O'Reilly & Associates, Inc. was aware of a trademark claim, the designations have been printed in caps or initial caps.

While every precaution has been taken in the preparation of this book, the publisher assumes no responsibility for errors or omissions, or for damages resulting from the use of the information contained herein.

This book is printed on acid-free paper with 85% recycled content, 15% post-consumer waste. O'Reilly & Associates is committed to using paper with the highest recycled content available consistent with high quality.

ISBN: 1-56592-382-0 [4/99]

Table of Contents



Preface

You are holding in your hands one half of *Bruce's Brain in a Book*. The other half of my brain is in the companion book, *Lingo in a Nutshell*. These books are the distillation of years of real-life experience with countless Director projects plus many hours spent researching and testing new features of Director 6, 6.5, and 7. While they can be used separately, they are ideally used as a single two-volume reference that costs less than most single Director books.

Director in a Nutshell focuses on the "concrete" aspects of Director—the Cast, the Score, Projectors, MIAWs, media (graphics, sound, digital video, and text), Director's windows, GUI components (buttons, cursors, menus), and Shockwave. *Lingo in a Nutshell* focuses on the abstract concepts in Lingo, such as variables, scripts, Behaviors, objects, mouse and keyboard events, timers, math, lists, strings, and file I/O.

If you already know a lot about Director or have been disappointed by the existing documentation, these are the books you've been waiting for. They address many of the errors and omissions in Macromedia's documentation and many third-party books. There is no fluff or filler here, so you'll miss a lot if you skim.

What Are These Books and Who Are They For?

Director in a Nutshell and *Lingo in a Nutshell* are Desktop Quick References for Director and Lingo developers who are familiar with Director's basic operation and need to create, debug, and optimize cross-platform Director and Shockwave projects. These books are concise, detailed, respectful of the reader's intelligence, and organized by topic to allow quick access to thorough coverage of all relevant information.

Because Lingo and Director are inextricably linked, I have kept all information on a single topic within a single chapter, rather than breaking it along the traditional Director versus Lingo lines (with the exception of Chapter 10, *Using Xtras*, in this book and Chapter 13, *Lingo Xtras and XObjects,* in *Lingo in a Nutshell*). Don't

assume that all the Lingo is consigned to *Lingo in a Nutshell*; *Director in a Nutshell* includes a lot of Lingo and you should be familiar with the Lingo basics covered in *Lingo in a Nutshell*.

This book (*Director in a Nutshell)* should not be confused with the third-party books that merely rehash the manuals; nor should it be considered an introductory book. It is exceptionally valuable for non-Lingo users but also covers Lingo related to those aspects of Director mentioned earlier. *Lingo in a Nutshell* covers both the basics of Lingo and its most advanced features. Each book covers both Windows and the Macintosh.

To describe these books as "beginner," "intermediate," or "advanced" would be misleading. Strictly as a comparison to other books on the market, you should consider their *coverage* extremely advanced, but the text itself is accessible to Director users of all levels. *Lingo in a Nutshell* allows Director users to take full advantage of Lingo's power, and *Director in a Nutshell* helps users of all levels deal confidently with the spectrum of Director's media types and features.

What These Books Are Not

These books are not a rehash of the Director manuals or Help system, but rather a complement to them, and as such are unlike any other books on the market.

These books are not a celebration of Director as multimedia Nirvana. They are for people who know that Director has many quirks and some bugs and want to know how to work around them quickly and effectively.

These books are not courses in graphic design, project management, Photoshop, HTML, or JavaScript. They will however help you integrate your existing skills and external content into Director's framework.

These books are not a Director tutorial, because I assume that you are familiar with the basics of Director's Cast, Score, Stage, and menus. They are not for people who need hand-holding. They are for people who can apply general concepts to their specific problem and want to do so rapidly.

These books are not perfect—errors are inevitable—so use them as a guide, not gospel. (These are the most thoroughly researched books ever written on Director and correct many errors and omissions in other sources.) While these books cannot anticipate all circumstances, they do provide the tools for you to confidently solve your specific problems even in the face of erroneous or incomplete information.

About This Book

Director in a Nutshell covers everything about content development and delivery in Director. It covers media and user interface elements and the Lingo to control them. It is divided into three major sections:

Part I, *Director's Core Components*

Chapter 1, *How Director Works*, explains Director's event-driven model and how it affects playback and screen imaging, and covers the hidden details of how the Score, Cast, and Lingo interact.

Chapter 2, *Being More Productive*, provides many tips and shortcuts to save you days over the course of a project, including details on hardware and software for development and testing and a primer on Windows and the Mac OS.

Chapter 3, *The Score and Animation*, covers animation techniques and optimization, the Score window and sprite manipulation, markers, and the Tempo channel. If you've had trouble adjusting to Director 6's new Score, this chapter is a gold mine. It also covers the Lingo for Score navigation, Score recording, and analyzing corrupted Score notation.

Chapter 4, *CastLibs, Cast Members, and Sprites*, covers all aspects of cast library management, importing assets into Director, linking to external media, and Cast window shortcuts. It also covers the Lingo for manipulating castLibs, cast members, and sprites, including comprehensive tables of supported media formats and all cast member and sprite properties for each asset type. It also includes several utilities to analyze and debug your Cast.

Chapter 5, *Coordinates, Alignment, and Registration Points*, covers Director's multiple coordinate systems (Stage-relative, monitor-relative, member-relative, and MIAW-relative) that determine sprite and window positioning. It also covers cast member registration points and Director's alignment tools. It tabulates the coordinate systems and units used by various Lingo keywords.

Chapter 6, *The Stage and Movies-in-a-Window*, covers the commands and operations that control the Stage and manipulate Movies-in-a-Window. It covers panning and scaling window views, communicating between windows, and setting window types and window properties.

Part II, *Delivery and Optimization*

Chapter 7, *Cross-Platform and OS Dependencies*, covers all cross-platform issues, including the differences in Lingo and Director amongst the Macintosh and various flavors of Windows.

Chapter 8, *Projectors and the Runtime Environment,* covers the options for creating runtime versions of your Director project for each platform. It also covers the Lingo to analyze various system properties at runtime, including determining the playback platform and the CD-ROM's drive letter. It also details differences between the authoring environment and Projectors.

Chapter 9, *Memory and Performance,* covers optimizing your project's performance and minimizing its memory usage. It details the memory and disk space required for each media type and lays out a memory budget for Director projects. It covers the Lingo that analyzes and controls memory allocation and cast member preloading, idle loading, purging, and unloading. It covers techniques to detect and fix memory leaks and to optimize all aspects of your project's performance.

Chapter 10, *Using Xtras,* covers installing and using Xtras in your Director projects. It describes in detail the Xtras that come with Director and tells you which ones you need to ship with your Projector and where to put them. See also Chapter 13 in *Lingo in a Nutshell*.

Chapter 11, *Shockwave and the Internet*, covers Shockwave delivery and creating linked CD-ROMs that access Internet-based content. It details which

Shockwave plug-ins are required for each browser on each platform, and covers the differences between Shockwave and standalone Projectors.

Part III, *Multimedia Elements*

Chapter 12, *Text and Fields*, covers the commands and operations for field and text cast members, including choosing the right type of text cast member and D7's new font cast members. See also Chapter 7, *Strings*, and Chapter 10, *Keyboard Events*, in *Lingo in a Nutshell*.

Chapter 13, *Graphics, Color, and Palettes*, covers the different types of graphical cast members and the Paint window. It includes a crucial explanation of palette management in Director, plus tips on solving palette problems. It also covers D7's new color model, vector shapes, and animated GIFs.

Chapter 14, *Graphical User Interface Components*, covers buttons, checkboxes, alert dialog boxes, cursors, and menus, and their control via Lingo. It also includes details on the Custom Cursor and Popup Menu Xtras.

Chapter 15, *Sound and Cue Points*, covers sound playback and manipulation, including *puppetSounds*, external sounds, Shockwave Audio (SWA), and cue points. It also covers sound mixing under Windows.

Chapter 16, *Digital Video*, covers video playback and manipulation via the Score and Lingo, including QuickTime and Video for Windows, plus details on QuickTime 3 and the QT3 Xtra.

Refer to *http://www.zeusprod.com/nutshell/appendices* for additional appendices on Flash, ActiveX, PowerPoint, Java, shipping checklists, and more.

Conventions Used in This Book

The following typographic, grammatical, and stylistic conventions are used throughout *Director in a Nutshell*.

The turkey icon designates a warning relating to the nearby text.

The owl icon designates a note, which is an important aside to the nearby text.

Typographical Conventions

- Lingo keywords (*functions*, *commands*, and *property names*) are shown in *italic*, except in tables, where they are only italicized when necessary to distinguish them from the surrounding text. Italic in tables usually indicates replaceable values.

- *`Arguments`*, *`user-specified`*, and *`replaceable`* items are shown in *`italic constant width`* and should be replaced by real values when used in your code.
- New terms are shown in *italic* and are often introduced by merely using them in context. Refer to *http://www.zeusprod.com/nutshell/glossary.html* for details.
- Options in dialog boxes, such as the *Tab to Next Field* checkbox, are shown in *italic*.
- Menu commands are shown as `MenuName` ➤ `MenuItem`.
- Constants such as `TRUE`, `FALSE`, and `RETURN` are shown in `Courier`.
- `#symbols` are preceded by the pound (`#`) character and shown in `Courier`.
- Optional items are specified with curly braces (`{}`) instead of traditional square braces (`[]`), which Lingo uses for lists. For example:

  ```
  go {to} {frame} whichFrame
  ```

  means that the following are equivalent:

  ```
  go whichFrame
  go to whichFrame
  go to frame whichFrame
  go frame whichFrame
  ```

- Allowed values for a property are separated by a vertical bar (`|`). The following indicates that *the checkBoxType* property can be set to 0, 1, or 2:

  ```
  set the checkBoxType = 0 | 1 | 2
  ```

Grammatical and Stylistic Conventions

- Most Lingo properties start with the word "the," which can lead to sentences such as, "The *the member of sprite property* can be changed at runtime." I often omit the keyword *the* preceding properties to make sentences or tables more readable, but you should include the "the" in your Lingo code.
- Lingo event handlers all begin with the word "on," such as *on mouseUp*. I often omit the word "on" when discussing events, messages and handlers, or in tables where the meaning is implied.
- Be aware that some Director keywords are used in multiple contexts such as the *on mouseUp* event handler and the *the mouseUp* system property. The intended usage is discernible from context and is stated explicitly only in ambiguous circumstances.
- I use terminology fairly loosely, as is typical among Lingo developers. For example a "*mouseUp* script" is technically "an *on mouseUp* handler within a script." The meaning should be clear from the context.
- I capitalize the names of Director entities, such as the Score, the Stage, the Cast, and the Message window. I don't capitalize general terms that refer to classes of items, such as sprite scripts.
- Most handler names used in the examples are arbitrary, although handlers such as *on mouseUp* that trap built-in events must be named as shown. I use variable names like *`myThing`* or *`whichSprite`* to indicate items for which you

should substitute your own values. When in doubt, see Chapter 18, *The Lingo Keyword and Command Summary*, in *Lingo in a Nutshell* or Director's online Help.

- I use few segues and assume you will re-read the material until it makes sense. As with a Dali painting, you must revisit the text periodically to discover details that you missed the first time.

Examples

- Example code is shown monospaced and set off in its own paragraph. If a code fragment is shown, especially using the `put` command, it is implicit that you should type the example in the Message window to see the result. Any text following "`--`" is the output from Director (shown in `constant width`), or a comment from me (shown in *`italic constant width`*):

```
set x = 5    -- Set the variable x to 5
put x        -- Display the value of x
-- 5
```

- Long lines of Lingo code are continued on the next line using the Lingo continuation character (¬) (created using `Opt-Return` or `Opt-L` on the Macintosh or `Alt-Enter` under Windows):

```
set the member of sprite (the currentSpriteNum) = ¬
   member "Hilighted Button"
```

- If you have trouble with an example, check for lines that may have been erroneously split without the Lingo continuation character (¬). Remember to use parentheses when calling any function that returns a value. Otherwise you'll either see no result or receive an error.

```
rollover         -- wrong
rollover()       -- wrong
put rollover     -- wrong
put rollover()   -- correct
```

- I sometimes use the single-line form of the *if...then* statement in an example for brevity. You should use multi-line *if...then* statements in your code. See Chapter 1, *How Lingo Works*, in *Lingo in a Nutshell* for details on the *if* statement.

```
-- This will usually work
if (x > 5) then put "It's True!"
-- But this is more reliable
if (x > 5) then
   put "It's True!"
end if
```

- If a handler is shown in an example, it is implied that the handler has been entered into the appropriate type of script. Unless otherwise specified, mouse event handlers such as *mouseUp* belong in sprite scripts, frame events handlers such as *exitFrame* belong in frame scripts, and custom utilities belong in movie scripts. I often show a handler followed by an example of its use. Type the handler into a movie script, and then test it from the Message window. If I

don't show a test in the Message window, either the handler does not output a visible result or it is assumed that you will test it yourself if you are interested:

```
-- This goes in a script, in this case a movie script
on customHandler
  put "Hello Sailor!"
end customHandler

-- This is a test in the Message window
customHandler
-- "Hello Sailor!"
```

- The output shown may vary inconsequentially from the results you see based on your system setup. Most notably, the number of decimal places shown for floating-point values depends on your setting for *the floatPrecision* property.
- If the output of a handler is extremely long, the results will not be shown in their entirety or may not be shown at all.
- The examples are demonstrative and not necessarily robust, and in them I assume that you provide valid inputs when applicable. It is good practice to include type checking and error checking in your actual Lingo code, as described in Chapter 3, *Lingo Coding and Debugging Tips*, and Chapter 1 in *Lingo in a Nutshell*. I often omit such checking to keep examples shorter and focused on the main issue.
- Some examples, particularly the tests performed from the Message window, are code *fragments*, and won't work without help from the studio audience. You should ensure that any variables required by the examples (particularly lists) have been initialized with meaningful values, although such initialization is not shown. For example:

  ```
  put count (myList)
  ```

  assumes that you have *previously* set a valid value for *`myList`*, such as:

  ```
  set myList = [1, 7, 5, 9]
  ```

- Some examples allude to text or field cast members, such as:

  ```
  set the text of field "Memory" = string(the freeBlock)
  ```

  It is implied that you should create a text or field cast member of the specified name in order for the example to work.
- Screenshots may not match your platform exactly.
- I present a simplified view of the universe whenever my assumptions are overwhelmingly likely to be valid. You can intentionally confuse Director by setting bizarre values for a property or performing malicious or unsupported operations, but you do so at your own risk. I cover situations where errors might occur accidentally, but you should assume that all statements presented as fact are prefaced by, "Assuming you are not trying to screw with Director just for fun . . . ." When necessary, I state my assumptions clearly.
- The myriad ways to perform a given task are shown when that task is the main topic of discussion, but not if it is peripheral to the subject at hand.

When incidental, I may show the clearest or most expedient method rather than the most elegant method.

- Following an example, I occasionally suggest ways to modify the code as a Reader Exercise. Solutions to Reader Exercises are posted at:

  http://www.zeusprod.com/nutshell/exercises/

- Examples are usually self-contained, but they may rely on custom handlers shown nearby. If an example builds on previous examples or material cross-referenced in another chapter, it is assumed that the relevant handlers have been entered in an appropriate script (usually a movie script).

New Features in Director 7

Director 7 is a great leap forward. There are no major changes to the Score or sprite messaging as in the D6 upgrade from D5, but there are many new features added on top of those in D6 and D6.5. For a complete list of new features, bugs, differences from D6, tips on updating movies from D6, and outstanding issues in both Director 7 and Shockwave 7, see the D7 FAQs starting at:

http://www.zeusprod.com/nutshell/d7faq.html

See Macromedia's summary of new features at:

http://www.macromedia.com/software/director/productinfo/newfeatures/

For documentation not available in the printed manuals or online Help, see:

http://www.macromedia.com/software/director/how/d7/

Select the *Fun* tab in the *About Director* window (under the Apple menu on the Macintosh or the Help menu under Windows) for demos of many new features including alpha channels, RGB colors, text and fonts, quads, rotation and skew, Flash 3, vector shapes, and animated GIFs.

If you need one or more of D7's new features, then upgrade. Regardless, take some time to learn D7 before creating a commercial product or upgrading a project from D6. The initial consensus is that D7 is extremely stable for a major revision. By the time you read this the D7.0.1 maintenance release should be available at *http://www.macromedia.com/support/director/upndown/updates.html*.

The Director 7 Shockwave Internet Studio includes these items which are not in the standalone Director upgrade:

- Behavior Library Palette (only limited Behaviors are included with standalone D7).
- Multiuser Server (Director for Windows includes the Windows server only, and Director for Macintosh includes the Macintosh server only, and you'll need the version that matches your web server. Linux and Unix versions are anticipated.)
- Macromedia Fireworks.
- Sound editor: Sound Forge XP (Windows) or Bias Peak LE (Macintosh).

Director 7 Features by Category

The major new features of Director 7 fall into several categories.

System architecture:

- D7 is based on a new playback engine first introduced as part of Shockwave 6.0.1, but completely different than the D6 engine. As such, it has many new features (especially dynamic sprite distortion), but also has new quirks.
- The Shockwave playback engine is now a system-level component (like QuickTime) that can be used by multiple browsers and so-called Slim Projectors. Slim Projectors can be under 200 KB and can even download missing components or Xtras from the Internet. Director 7 and Shockwave 7 continue the trend towards modularization by using many Xtras, which you can omit if the feature is not needed.
- The underlying engine is the same in all versions of Shockwave 7 for all browsers, Director 7 on both Macintosh and Windows, plus the new ShockMachine (a local Shockwave player). Expect to see fewer differences across playback platforms than in prior versions. Any playback environment can adopt Shockwave's security hobbles by declaring itself as a "safe" environment by setting *the safePlayer* to `TRUE`.

Score, animation, authoring, and playback improvements:

- The Stage is a standard window that can be closed or moved during authoring, or placed in front of all MIAWs (D7.0.1 fixes a bug in this regard).
- New sprite properties and media types create eye-popping animation with minimal cast members (ideal for Shockwave delivery).
- Colorize, skew, rotate, and mirror bitmaps, Flash, animated GIFs, text, and vector shapes on Stage or using the Sprite Toolbar and Sprite Inspector.
- Quad distortion performs 3D-like effects at runtime on text, bitmaps, and animated GIFs. Reverse the corners to see the "back" of a sprite or twist it into a bowtie.
- Up to 1000 sprite channels and 999 frames per second playback.
- Dynamic z-ordering of sprite channels via Lingo (the *locZ of sprite* property).
- Alpha channels (partial transparency) and runtime dithering.
- Multiple monitors supported under both Windows and Macintosh.
- The Paint window supports 16-bit and 32-bit painting.
- Dynamic selection of sound mixer, including QT3 Mixer, under Windows.
- Improved ink effects, sprite colorization, and blend. True RGB color model allowing colorizing of sprites in all color depths.
- Capture the Stage into a cast member using *the picture of the stage*, or crop it with the new *crop()* command.
- Improved grid snapping that uses the nearest corner or side instead of the registration point to snap a sprite to the grid.

Media improvements and additions:

- D7 includes all the import and export media features added in D6.5 including QuickTime 3, Flash 2, ActiveX, Java Export, PowerPoint import, and custom animated cursors, plus new support for Flash 3, MPEG 3, and improved QuickTime 3 support.
- New animated GIF members, plus JPEG and GIF import, and support for internal compressed JPEG, GIF, and animated GIF assets.
- Text cast members allow anti-aliased text to be edited, rotated, skewed, and colorized at runtime. Some support for hypertext links, HTML import, and RTF styles, including superscripts and subscripts. Text, field, and script cast members are no longer limited to 32 KB.
- Compressed font cast members that can be used by both text and fields to provide platform-independent fonts without requiring font installation.
- Programmable vector shapes for dynamic Bézier curves, charts and graphs, splines, and polylines.
- PhotoCaster Lite (which allows import of separate Photoshop layers) and a demo version of the Beatnik sound Xtra are included.

Lingo improvements include:

- Dozens of new Lingo commands (see *http://www.zeusprod.com/nutshell/d7lingo.html*).
- Scripts no longer limited to 32 KB.
- Improved *traceLoad* features and new *getStreamStatus()* function.
- Debug MIAWs in the D7 debugger.
- Lingo script colorization (I don't like it, personally).
- Library Palette provides many built-in Behaviors (included in Director 7 Shockwave Internet Studio only).
- Improved timers and Y2K-compliant date functions.

D7 supports streamlined JavaScript-like dot notation (a.k.a. dot syntax). Dot syntax is a shorthand way to specify member and sprite properties. It is available in most situations, and doesn't require the keyword `set`. For example:

```
sprite(5).loc = point (50, 100)
member(2, 3).directToStage = TRUE
```

can be used instead of:

```
set the loc of sprite 5 = point (50, 100)
set the directToStage of member 2 of castLib 3 = TRUE
```

D7's new bracket syntax is useful with lists. For example:

```
x = exampleList[1]
someList[7][4] = "newValue"
```

can be used instead of:

```
x = getAt (exampleList, 1)
setAt (getAt(someList, 7), 4, "newValue")
```

For many more examples and details, see Chapters 4 and 12, and *http://www.zeusprod.com/nutshell/dotsyntax.html*.

Shockwave 7 and Internet-related improvements:

- Shockwave 7 (SW7) uses a single system player and *Xtras* folder even if using multiple browsers. Automatic incremental upgrades of Shockwave 7 components (smaller downloads). A progress bar now appears to indicate movie downloading status.
- Automatic downloading of digitally signed Xtras and improved security against potentially damaging Xtras in Shockwave
- More convenience: Preview in Browser and a built-in Web 216 (browser-safe) palette. Improved AfterShock (although animated GIF export was dropped). Better streaming management, including *getStreamStatus()*. Support for web standards (HTTPS, XML, simple text HTML tags including tables, post FORM data with *postNetText*, and Java export).
- Multiuser Server (included with the Director Studio only) can create multiplayer games, chat rooms, and shared on-line databases. The Multiuser Xtra also allows peer-to-peer connections.
- ShockMachine is an enhanced player offering the ability to save and play Shockwave movies locally, with full screen playback, volume controls, and custom caching, without requiring a browser.

What's Missing in Director 7

Director 7 has a boatload of new features, but the following were dropped since D6, or not added, though widely hoped for, in D7:

- Macromedia's *Learning Lingo* manual has been incorporated into the *Using Director* manual. Many of the new features are documented on-line only (see URL cited earlier). D7's help system is no longer context-sensitive, but this may be fixed in D7.0.1.
- There is no native ability to render common HTML tags beyond limited support for HTML in text members. You still need an Xtra to "put a browser inside Director."
- No improvements have been made to Director's ability to handle DVD and MPEG video formats since version 6.0. The support for DVD is limited, but can be augmented with the DirectMedia Xtra from Tabuleiro da Baiana (*http://www.tbaiana.com*).
- There is still not support for random access to SWA files. Macromedia justifies this by saying that most SWA files are streamed from the internet and therefore random access is impractical. Use QT3 movie audio tracks, which can be accessed randomly, instead.
- There is no easy way to permanently attach multiple Behaviors with custom properties to a sprite via Score Recording, although the new *scriptList of sprite* property provides read-only access to attached Behaviors and their current properties.

- D6 rich text is obsolete and has been replaced by D7 text members.
- QuickTime 2 is not supported. QuickTime 3 is required, although Video for Windows AVI files are still supported under Windows.
- SoundEdit 16 has been replaced by Bias Peak LE in the Macintosh Studio package. Extreme 3D and xRes have been supplanted by Fireworks.
- D7 does not support 68K Macs (requires a PPC or G3, and Mac OS 7.5.3 or higher) or Windows 3.1 (requires Windows 95/98/NT and a Pentium).
- RSX/DirectSound sound mixing is not supported in D7 as it was in D6, but D7.0.1 includes a DirectSound mixer that doesn't require RSX.
- No improvements or additions have been made to D7's project management capabilities. There is still no source code or version control system and no improved tools for collaboration among multiple developers.
- The widely rumored spell-checker and encryption Xtras have yet to surface.

Director Resources

The best thing about Director is the extended community of developers that you can torment for assistance. This book notwithstanding, Director is 90% undocumented. Visit Macromedia's web site frequently, and plug into the broader Director community via mailing lists and newsgroups.

Online Resources

The following resources are mandatory for serious Director developers. Links to additional URLs cited throughout this book can also be found at *http://www.zeusprod.com/nutshell/links.html*.

Director in a Nutshell and Lingo in a Nutshell

O'Reilly and Associates:

http://www.oreilly.com/catalog/directnut/
http://www.oreilly.com/catalog/lingonut/

Example code, bonus chapters, links to all URLs in the books:

http://www.zeusprod.com/nutshell

Web Review—all things browser- and web-related:

http://www.webreview.com/

Macromedia

Macromedia home page and mirror sites:

http://www.macromedia.com
http://www-euro.macromedia.com
http://www-asia.macromedia.com

Director 7 new features, upgrade policy, and online docs:

http://www.macromedia.com/support/director/how/d7/
http://www.macromedia.com/software/director/productinfo/newfeatures/
http://www.macromedia.com/software/director/upgrade/

Director 7.0.1, D6.5 Service Pack for Windows and other updaters:

http://www.macromedia.com/support/director/upndown/updates.html

Director Developers Center (searchable database of tech notes and tips):

http://www.macromedia.com/support/director/
http://www.macromedia.com/support/sdirector/ts/nav/
http://www.macromedia.com/support/director/how/subjects/

Shockwave Developer Center:

http://www.macromedia.com/shockwave/
http://www.macromedia.com/support/director/how/shock/

Dynamic HTML and Shockwave:

http://www.dhtmlzone.com/swdhtml/index.html

Director-related newsgroups:

http://www.macromedia.com/support/director/interact/newsgroups/
news://forums.macromedia.com/macromedia.plug-ins
news://forums.macromedia.com/macromedia.director.basics
news://forums.macromedia.com/macromedia.director.lingo

Priority Access (fee-based) technical support:

http://www.macromedia.com/support/techsupport.html
http://www.macromedia.com/support/director/suprog/

Beta program:

http://www.macromedia.com/support/program/beta.html

Director feature suggestions:

mailto:wish-director@macromedia.com

Phone support:

MacroFacts (fax information): 800-449-3329 or 415-863-4409
Technical support: 415-252-9080
Main Operator: 415-252-2000

User groups:

http://www.macromedia.com/support/programs/usergroups/worldwide.html

Developer Locator (find a Director or Lingo developer in your area):

http://www.macromedia.com/support/developer_locator/

Macromedia User Conference (UCON) May 25–27, 1999, in San Francisco, CA:

http://ucon.macromedia.com

Web Sites and Xtras

Zeus Productions (my company) technical notes and Xtras:

http://www.zeusprod.com

UpdateStage (monthly technical articles and the Director Quirk List and Xtras):

http://www.updatestage.com
ftp://ftp.shore.net/members/update/

Director Online Users Group (DOUG)—articles, interviews, reviews:

http://www.director-online.com

Maricopa Director Web (the mother ship of Director information):

http://www.mcli.dist.maricopa.edu/director/tips.html
ftp://ftp.maricopa.edu/pub/mcli/director

Lingo Behavior Database (example Behaviors):

http://www.behaviors.com/lbd/

Links to additional third-party web sites:

http://www.mcli.dist.maricopa.edu/director/net.html
http://www.macromedia.com/support/director/ts/documents/tn3104-dirwebsites.html

Third-party Xtras:

http://www.macromedia.com/software/xtras/director

FMA Online (links to many Xtra developers):

http://www.fmaonline.com

Xtras developer programs:

http://www.macromedia.com/support/program/xtrasdev.html
http://www.macromedia.com/support/xtras.html

Apple QuickTime and developer sites:

http://developer.apple.com
http://quicktime.apple.com

Mailing Lists

If you have the bandwidth, these mailing lists are often useful resources for Director, Shockwave, Xtras, and Lingo questions (see the Macromedia newsgroups listed earlier). These mailing lists generate a *lot* of email. Subscribe using DIGEST mode to avoid hundreds of separate emails each day.

DIRECT-L (Director and Lingo):

Send the following in the body (not subject) of an email to *listserv@uafsysb.uark.edu*:

```
SUBSCRIBE DIRECT-L yourFirstName yourLastName
SET DIRECT-L DIGEST
```

Archives: *http://www.mcli.dist.maricopa.edu/director/digest/index.html*
MailList: *http://www.mcli.dist.maricopa.edu/director/direct-l/index.html*

Lingo-L (Lingo):

http://www.penworks.com/LUJ/lingo-l.cgi

ShockeR (Shockwave):

Send the following in the body of an email to *list-manager@shocker.com*:

```
SUBSCRIBE shockwave-DIGEST yourEmail@yourDomain
```

Archive: *http://ww2.narrative.com/shocker.nsf*
MailList: *http://www.shocker.com/shocker/digests/index.html*

Xtras-L (Xtras for Director):

Send the following in the body of an email to *listserv@trevimedia.com*:

```
SUB XTRAS-L yourFirstName yourLastName
```

Flash Resources

Flash newsgroup:

news://forums.macromedia.com/macromedia.flash

Flasher mailing list:

Send the following in the *body* of an email to *list-manager@shocker.com*:

```
SUBSCRIBE Flasher yourEmail@yourDomain
```

Flash Pad:

http://www.flasher.net/flashpad.html

Flash discussion group:

http://www.devdesign.com/flash

We'd Like to Hear from You

We have tested and verified all of the information in this book to the best of our ability, but you may find that features have changed (or that we have made mistakes). Please let O'Reilly know about any errors you find by writing:

O'Reilly & Associates, Inc.
101 Morris Street
Sebastopol, CA 95472
800-998-9938 (in U.S. or Canada)
707-829-0515 (international/local)
707-829-0104 (fax)

You can also send us messages electronically. To be put on the mailing list or request a catalog, send email to:

nuts@oreilly.com

To ask technical questions or comment on the book, send email to:

bookquestions@oreilly.com

Dedications

Director in a Nutshell is dedicated to Zoë, who likes the ostrich on the cover; to Ariel, who has been waiting most of her life for me to finish this book; to Zachary, who has been waiting his *entire* life for me to finish this book; and to Mildred Krauss, the most literate, intelligent, and sincere person I've had the good fortune to be related to.

In memoriam

I wish to acknowledge the passing of my great-uncle Mark Daniel. It is with great personal sadness that I mourn his departure from the world into which, as the family obstetrician, he brought me and my siblings. May those who knew and loved him take comfort in the lives that he touched while he was here.

Acknowledgments

I am indebted to many people, some of which I've undoubtedly omitted from the following list. Please buy this book and recommend it to friends so that I can thank the people I've forgotten in the next revision.

My deep appreciation goes out to the entire staff at O'Reilly, whose patience, professionalism, and unwavering dedication to quality are directly responsible for the existence and depth of this book. Special thanks goes to my editors Katie Gardner and Troy Mott, who I put through heck if not hell, for their tolerance and perseverance; to the series editor Tim O'Reilly for recognizing the genuine article; to Edie Freedman, whose choice of an ostrich that looks like me for the cover made my wife amorous; and to Seth Maislin, for his index par excellence. My thanks also to Sheryl Avruch, Frank Willison, Robert Romano, Mike Sierra, and the O'Reilly production staff, including Clairemarie Fisher O'Leary, Nicole Gipson Arigo, Ellie Cutler, and Jane Ellin, who turn a manuscript into a book; to the sales and marketing staff, who bring home the bacon; and to all the O'Reilly authors in whose company I am proud to be.

I must especially thank Nancy Kotary, my production editor, for her tireless and heroic efforts on this book. Nancy is truly the epitome of what an editor should be—an invisible hand that improves a manuscript without detracting from the author's voice or content. I credit Nancy with turning me from a writer into a true author.

This project would not have happened without the efforts of my agent, David Rogelberg of Studio B Productions (*http://www.studiob.com*). He was instrumental in the development and genesis of both *Director in a Nutshell* and *Lingo in a Nutshell*, for which I am forever grateful. My thanks also to Sherry Rogelberg and to the participants of Studio B's Computer Book Publishing list (particularly John Levine).

The quality of the manuscript reflects my excellent technical reviewers, all of whom made time for this semi-thankless job despite their busy schedules: Lisa Kushins, who verified items to an extent that astounded me and provided feedback that improved every chapter she touched; Hudson Ansley, whose keen eye and unique perspective also improved the book immeasurably; and Mark Castle

(*http://www.the-castle.com*), who helped shape the style and content from the earliest stages. My thanks also goes out to all my beta readers, who provided useful feedback, particularly Roger Jones, John Williams, Ted Jones, and Alex Zavatone, and to the reviewers who were kind enough to peruse the manuscript and offer the choice quotes you'll find on the back cover.

I can not begin to thank all the Macromedians who develop, document, and support Director, many of whom provide technical support on their own time on various mailing lists. My special thanks goes to Buzz Kettles, for all his feedback regarding Shockwave audio and sound mixing. My thanks again to Lalit Balchandani, David Calaprice, Jim Corbett, Landon Cox, Ken Day, Peter DeCrescenzo, David Dennick, John Dowdell, Mike Edmunds, John Embow, Eliot Greenfield, Jim Inscore, David Jennings, James Khazar, Leona Lapez, S Page, Andrew Rose, Joe Schmitz, Bill Schulze, Michael Seery, Werner Sharp, Karen Silvey, Gordon Smith, Joe Sparks, John Thompson, Karen Tucker, John Ware, Eric Wittman, Doug Wyrick, and Greg Yachuk, all of whom fight the good fight on a daily basis. A special thanks to Stephen Hsu of Puma Associates, for the use of his equipment. My thanks goes out to the wider Director community many of whom I thanked in *Lingo in a Nutshell*, and to Jeff Buell, Kurt Cagle, Marc Canter, Chino, Jamie Ciocco, Jim Collins, Rob Dillon, Greg Griffith, Colin Holgate, Marvyn Hortman, Richard Hurley, Jeremy Scott Knudsen, Brian Kromrey, Renfield Kuroda, George Langley, James Newton, John Nyquist, Daniel Plaenitz, Andrew Rose, Gary Rosenzweig, Terry Schussler, Brian Sharon, John Taylor, Michael Weinberg, Mark Whybird, and Charles Wiltgen, whom I did not.

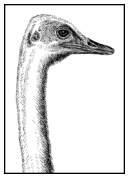
I still owe a debt of gratitude to Professor David Thorburn, who taught me more about writing than anyone before or since. Please send any complaints to him.

I want to acknowledge both my immediate and extended family, especially my parents (you know who you are), whose love and encouragement molded me into a reasonable facsimile of an adult; and to my wife Michele, whose love and encouragement made these books possible.

I'd like to thank you for taking the time to read this book. It is not a static lecture, but an ongoing conversation between you the reader and me the author. Feedback from many customers, clients, and friends has already shaped its content and, with any luck, will shape many future revisions. Let us see if we can learn some things about Director and something about ourselves in the process.

—Bruce A. Epstein
Franklin Park, N.J., March 1998

"Wisdom consists of knowing when to avoid perfection."
—Confucius



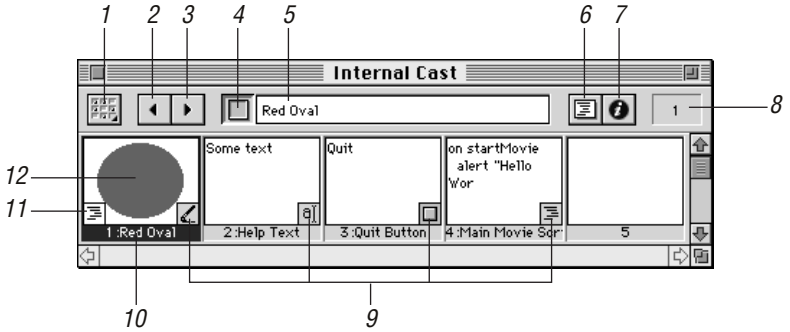
CHAPTER 4

CastLibs, Cast Members, and Sprites

This chapter covers importing assets, using the Cast window, and the Lingo that manipulates castLibs, cast members, and sprites. If you are unfamiliar with sprites and cast members, refer to the tutorials in Macromedia's *Using Director* manual.

Cast Libraries

Director assets are stored as cast members within *castLibs* (cast libraries, or simply *casts*). The Cast window is shown in Figure 4-1.



1. Castlib selector
2. Previous cast member
3. Next cast member
4. Drag cast member tool
5. Member name
6. Open script window
7. Open cast member properties
8. Member number
9. Cast member icons
10. Selection highlight
11. Cast script indicator
12. Thumbnail

Figure 4-1: Cast window

Prior to Director 5, cast members were referred to using the *cast* keyword (which, though obsolete, is still supported for backward compatibility). In conversation, the word *cast* refers to a castLib, not an individual member, but Lingo uses the keyword *castLib* to refer to cast libraries and the keyword *member* to refer to members (i.e., cast members) within a cast library.

Director supports both internal and external castLibs. A movie always contains at least one internal castLib, which may have zero cast members. You can optionally create additional internal castLibs, which are private to a single Director movie (although a MIAW can access the main movie's cast using *tell the Stage*). External castLibs are often linked (attached) to one or more movies, but they can also be used as standalone libraries during authoring (so-called "floating" castLibs).

An *internal* (*unlinked* or *embedded*) cast member is one in which the data is incorporated directly into the Cast and stored in Director's internal format for the given data type. For example, text cast members are always embedded. If an asset has been imported as an unlinked cast member, you do not need to distribute the original asset file with your Projector, but it should be kept in case you need to modify it and reimport it.

A *linked* cast member is one that points to an external file containing the data of interest. Some cast members, such as digital videos, are always linked.

The external asset files associated with linked cast members must be distributed with your Projector.

Some cast members—notably sounds and bitmaps—can be either linked or unlinked. Don't confuse a linked (external) castLib with linked cast members (which can reside in either internal or external castLibs).

You can sometimes access external assets without creating a cast member. The *sound playFile* command will play an arbitrary external WAVE or AIFF file. Some Xtras also access external files without necessarily creating a new cast member. The FileIO Xtra can read an external text file. External assets can be accessed dynamically by changing a cast member's *fileName of member* property to point to a new file.

Multiple and External CastLibs

You can link (attach) one or more external castLibs into your movie and open multiple Cast windows to view them simultaneously. External castLibs are convenient for holding assets that are used in more than one movie. You can use multiple internal or external castLibs to organize assets such as graphics, sounds, and scripts.

Any asset used in more than one movie should be stored in an external castLib. This eases maintenance, reduces storage requirements, and ensures consistency across movies. Keeping common scripts in an external castLib eases testing, editing, and debugging.

If you drag a cast member between two castLibs (either internal or external) that are linked to the same movie, it is moved (not copied) from the original castLib to the destination castLib.

All external castLibs need not be linked to the current movie. Use `File ➤ New ➤ Cast` to create a new external castLib and `File ➤ Open` to open an existing unlinked external castLib. Unlinked castLibs do not appear in the Cast window castLib selector and are not accessible via Lingo, but dragging a cast member from an unlinked external castLib to another castLib will copy it to the destination castLib.

If you place a cast member from an unlinked external castLib onto the Stage or into the Score, Director prompts you to link the castLib to the current movie or to copy the cast member to one of the castLibs already attached. If the unlinked castLib's names contain the word "Library" (such as the D6 Behavior Library), Director automatically copies cast members to the first internal castLib without prompting.

You can repeatedly drag Behaviors from any unlinked external castLib (such as the Behavior Library) directly to the Score; only one copy of the Behavior will be copied to your internal cast. (Director uses a unique internal ID number to prevent duplicate copies of a single Behavior.) Any modification to the Behavior's script or its cast member name will cause Director to import a fresh copy the next time the Behavior is applied.

The D6 Widget Wizard uses Score Recording and often inserts multiple copies of the same bitmaps and Behaviors into your internal cast. Apply Behaviors by hand as per the Widget Wizard's help instructions to avoid rampant duplication when using the same widget multiple times.

In D7, the Library Palette Window replaces `Xtras ➤ Behavior Library`. Add your own Libraries to the Library Palette by placing castLibs containing Behaviors in the *Xtras/Libs* folder or one of its subfolders. See the many useful existing Behaviors in the Library Palette that comes with the Director Multimedia Studio (but not the standalone version).

Great uses for external castLibs

There are numerous reasons to use external castLibs, even when an asset is not used in multiple movies:

Collaboration

By placing the different assets in different castLibs, multiple developers can work on the same project semi-independently. An artist can update graphics and deliver them in a new external cast library, or a sound designer can provide replacement sounds.

Smaller backups and downloads

By separating assets in external castLibs, you can back up only the data that has changed. The time and disk space savings can be significant. If collaborating remotely, you need not upload 10 MB of graphics and sounds to change the Lingo.

Use caution when moving cast members within an external castLib. Although Director will try to update the movie's Score to reconcile changes in the castLib, it is safest to tell collaborators not to move any cast members in an external castLib.

Internationalization of multilanguage versions

Store text, field, and bitmap cast members that need to be translated to different languages into an external castLib. Place the translated assets into the same cast member positions as the originals. Don't forget culture-specific images, such as mailboxes, police, taxicabs, and flags, and beware of items that might offend local users. For example, in some countries, a "thumbs-up" sign is an obscenity equivalent to the middle finger in the U.S. (Note that the Macintosh "counting fingers" animated hand cursor never shows the thumb up by itself. It starts and ends with the innocuous pinkie.)

Source code security

If you are a consultant, you can keep your Lingo scripts in a protected external castLib. You can withhold the source code permanently or until you've been paid without otherwise hindering delivery and testing.

Pseudo-editing multiple movies

Placing scripts, graphics, or sounds in an external castLib makes it easy to edit related items used in different movies without switching between movies. Although Director can't open two movies at once, you can open external castLibs from different movies and edit them simultaneously.

Script, asset, and Behavior libraries

You can create external castLibs of utility scripts (such as those in this book), common sounds (mouse-clicks and your company jingle), and graphics (your company logo). If you place your castLib in your *Xtras* folder, it will appear under the `Xtras` menu. Give the castLib a name containing the word "Library" (with or without a .CST extension) and Director 6 will copy assets from it without prompting. In D7, use the *Xtras/Libs* folder.

Disadvantages of external castLibs

External castLibs have their limitations:

Assets are not necessarily stored in the optimal order

When you use `File` ➤ `Save and Compact` on a Director movie, the cast members in any internal castLibs are stored in the order in which they are used in the Score. When compacted, cast members in external castLibs are stored in the order in which they appear in the Cast window.

Because external castLibs are usually accessed by multiple movies' Scores, there is no single optimal storage order. Rearrange them manually to improve load times or use `Modify` ➤ `Sort` ➤ *Usage in Score*.

Limited number of castLibs

In theory, you can attach an unlimited number of internal and external cast libraries to a Director movie. The 16-bit version of Director 5 for Windows 3.1

and 16-bit Projectors were limited to 12 external castLibs. This limit was removed in Director 6, but the maximum number of file handles under Windows 3.1 is set by the *CONFIG.SYS* file, not by Director. Avoid an excessive number of castLibs (more than six or so). Even in D7, an inordinate number of castLibs slows a movie's startup.

Potential conflicts in Lingo

System event handlers (such as *startMovie*, *idle*, *exitFrame*, *mouseDown*, *mouseUp*, and *keyDown*) within movie scripts in an external castLib might be called unintentionally for any movie to which that castLib is linked. Likewise, duplicate handler names in movie scripts of multiple castLibs will conflict.

Potential conflicts in cast member names and references

Having two cast members with the same name would prevent you from referring to the second one by name. When using multiple castLibs, cast member references should include the castLib, such as:

```
member whichMember of castLib whichCast
```

or in D7 notation:

```
member(whichMember, whichCast)
```

Collaboration must be undertaken with caution

If you change the Score while someone else is changing an external castLib you must reconcile the file versions at some point. See "Adjusting Score references to external cast libraries" later in this chapter.

Memory leaks and bugs

Some bugs occur only when using an asset in an external castLib. For example, there have been problems with sounds in external castLibs not being released from memory and occasional problems with Xtra cast members in external castLibs. If you encounter what seems like an obscure problem, try moving the asset to an internal castLib. (Also upgrade to the latest version of Director. Director 6.0.1 fixed a problem with moving film loops between castLibs in D6.0, and D7.0.1 fixes problems with fonts in external casts.)

Unlinking and relinking external castLibs

If you use `Modify ➤ Movie ➤ Casts ➤` *Remove* to unlink a castLib, Director will ask you whether to remove all Score references to it. This indicates that the movie uses some of the members stored in the castLib and you should cancel the operation. If your goal is to replace an external castLib, you can set *the fileName of castLib* property or use `Modify ➤ Cast ➤ Properties` to modify the link. Alternatively, you can close the movie and then move or delete the external castLib. When you reopen the movie, Director will prompt you to locate the missing castLib, allowing you to specify a different castLib.

If the Score uses cast members from an external castLib that is replaced, the members in the replacement castLib must use the same *memberNum* positions as in the original castLib.

To export an internal castLib to an external castLib file, use *save castLib* and specify an external file as the destination.

There is no documented Lingo to create and attach a new castLib at runtime. You can create and attach a dummy castLib during authoring and reassign its *fileName of castLib* property as needed. The following unsupported Lingo works in most cases:

```
importFileInto findEmpty(1), "myFile.cst"
```

Search the Direct-L archives for the phrase "importFileInto castLib" for details and caveats.

The freeware CastControl Xtra will attach and detach castLibs at runtime (*http://www.magna.com.au/~farryp/director/xtras/*).

```
-- The first parameter sets the name of castLib property.
-- The second sets the fileName of castLib property.
AttachCastLib internalName, filePath
```

CastControl can detach a castLib by number or by name, but don't attempt to detach internal castLibs (attempting to detach the first internal castLib crashes):

```
detachCastLib 3
detachCastLib the name of castLib 3
```

The CastEffects Xtra (*http://www.penworks.com/xtras/castfx*) can also create and link a new castLib dynamically at runtime (and it can scale, rotate, and extract images at runtime). Also see the Effector Set Xtra (*http://www.medialab.com*), which can transform (scale, rotate, etc.) cast members at runtime, although D7 adds native support for these features.

Shared Cast versus external cast libraries

In Director 3 and 4, only a single external castLib known as the *Shared Cast* was allowed. The Shared Cast was actually a standard Director file whose cast members were accessible from the main movie. The main movie would automatically look for a Shared Cast file in the same folder (there was no explicit link between the main movie and the Shared Cast). To use separate Shared Casts, you needed to place movies in different folders. Members in the Shared Cast appeared at the end of the main movie's Cast window and were distinguished by italicized names and numbers. To prevent conflicts, the cast members in the Shared Cast had to use cast member slots after those used by the main movie(s).

Director 5 and later support multiple external castLibs that are explicitly linked into a movie and can reside anywhere. (In Director 6 and 7, external castLibs can even reside on the Internet.)

When updating a movie from D4 to D6, the Shared Cast is renamed *SHARED.CST* and linked as an external castLib to all movies updated in the same batch. In D4, the Shared Cast's cast members always used the same cast member slots. To simulate this when updating to D6, *the number of member* of the first cast member slot of castLib 2 (presumably the new *SHARED.CST* file) is *kludged* (rigged) to coincide with the first used cast member number in the old D4 *SHARED.DIR* (Shared Cast) file. If the old D4 Shared Cast "started" at cast member 100, when updated to D6, *the number of member 1 of castLib 2* reports 100, not 131073, as it would for

movies created from scratch in D6. Furthermore, this holds true for whichever castLib is castLib 2 in the updated movie, even if it is not *SHARED.CST*! This can wreak havoc if *SHARED.CST* is not the second castLib.

In D5 and later, the castLib number of any external castLib will depend on the number of internal castLibs and the order in which external castLibs are attached. The number of the first external castLib is always one greater than the number of internal castLibs. Table 4-1 outlines the use of the Shared Cast or external castLibs in Director.

Table 4-1: Shared Cast Versus External CastLibs

Version	External CastLib Name	Notes
Director 3.1.3	Shared Cast (Mac), *SHRDCST.DIR* (Windows)	File format was not cross-platform. Shared Cast resided in same folder with main movie. D6 will not update from D3.1.3.
Director 4	*SHARED.DIR* (unprotected) or *SHARED.DXR* (protected)	Cross-platform file format, but Shared Cast still resided in same folder with main movie. D7 will not update from D4.
Director 5	*<AnyName>.CST* (unprotected), *<AnyName>.CXT* (protected), or *<AnyName>.CCT* (Shockwave)	Multiple external castLibs allowed. CastLibs can use any name and can reside in any folder.
Director 6 and 7	Same as Director 5.	Shockwave casts can be used locally or reside at any URL.

Adjusting Score references to external cast libraries

Moving, adding, or deleting cast members in an external castLib affects all Director movies that link to that castLib, including ones that are not open when the changes occur. Director tracks these changes via a cast member's unique internal ID. It prompts you to update the Score references the next time you open a movie using that external castLib (as shown in Figure 4-2) even if the altered cast members in the external castLib are not used in the current movie.

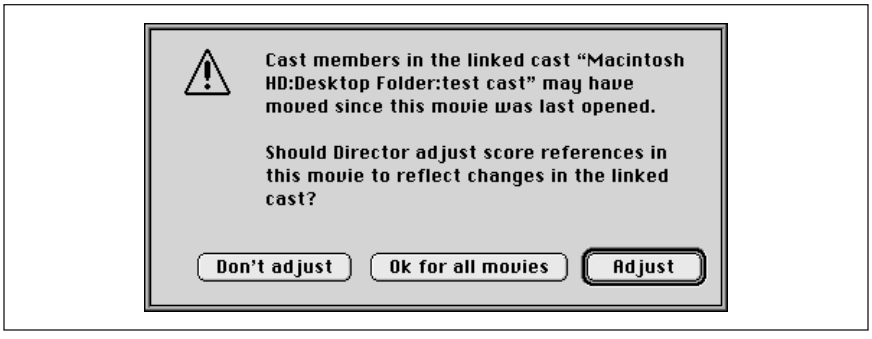


Figure 4-2: Adjusting references to linked castLibs

The three possible responses to the dialog box are not particularly intuitive:

Adjust
: Adjusts all Score references to accommodate any moved cast members. This is the default option and usually the correct choice. If a cast member has been deleted, Director does not remove the Score reference but will point to a nonexistent cast member unless the cast member slot is reused.

Don't adjust
: Leaves the Score alone. This is a dangerous option, because it is likely that the sprite references in the Score will point to wrong or nonexistent cast members. If you choose this accidentally, quit Director without saving the current movie, and then reopen the movie. If you really don't want to adjust the Score references, you must save the movie after hitting this option to prevent being warned again. You'll need to make some other change to enable the `File➤Save` option, or use `File➤Save As` or `File➤Save and Compact` instead.

Ok for all movies
: This is the vaguest prompt of all time. In Director 4, this option was named *Don't Warn Me Again*. It does *not* adjust the Score and prevents Director from warning you if other movies using the same external castLib need to be updated. Use this only if you added cast members to an external castLib but did not move or delete any cast members. If you choose this by mistake, quit Director without saving the current movie.

Cast Library Mechanics

Table 4-2 shows the commands that manage internal and external cast libraries.

Table 4-2: Working with CastLibs

Operation	**Menu Command**	**Mac**	**Win**
Open/Close Cast window	Window ➤ Cast	Cmd-3	Ctrl-3
Open more Cast windows	Window ➤ Cast ➤ *CastLibName* Window ➤ New Window	Opt-click castLib pop-up	Alt-click castLib pop-up
Open unlinked external Cast	File ➤ Open or use Toolbar button.	Cmd-O	Ctrl-O
Open Behavior Library (D6)	Xtras ➤ Behavior Library	None	None
Open Java Behaviors (D6.5)	Xtras ➤ Behavior Library for Java	None	None
Open Library Palette (D7)	Window ➤ Library Palette or use Toolbar button	None	None
Create, link, remove, or modify castLibs in use	Modify ➤ Movie ➤ Casts	Cmd-Shift-C	Ctrl-Shift-C

Table 4-2: Working with CastLibs (continued)

Operation	Menu Command	Mac	Win
Create a new castLib	File ➤ New ➤ Cast Modify ➤ Movie ➤ Casts ➤ New Choose *New Cast* from Cast window castLib pop-up or use Toolbar button.	Cmd-Opt-N	Ctrl-Alt-N
Switch castLib displayed in a Cast window	Use *Choose Cast* pop-up in Cast window.	Cmd-↑ Cmd-↓	Ctrl-↑ Ctrl-↓
Link to existing castLib on local drive on Internet	File ➤ Import ➤ Director Cast Modify ➤ Movie ➤ Casts ➤ Link Choose Internet button in file selection/ import dialog box.	Cmd-R Cmd-Shift-C	Ctrl-R Ctrl-Shift-C
Change castLib fileName, name, or preLoadMode	Modify ➤ Cast Properties Modify ➤ Movie ➤ Casts ➤ Properties	Ctrl-click in Cast window	Right-click in Cast window
Change preferences for cast member thumbnails	File ➤ Preferences ➤ Cast or Context pop-up in Cast window	None	Alt-F, F, C
Sort cast members	Select cast members, then choose Modify ➤ Sort.	None	None

Compacted, protected, and compressed castLibs

There are three "formats" for both Director movies and castLibs:

Standard format (including "Compacted" movies)

The standard Director formats are the well-known movie (DIR) and castLib (CST) files used primarily during authoring. They can also be used with a Projector (if left external rather than embedded in the Projector) and even played locally via a Shockwave-enabled browser. *Compacted* movies and castLib files are different only in that compacting removes any deleted cast members and optimizes cast member order and Score notation. Compacting a file does not protect or compress the assets beyond Director's native cast member compression.

Protected format

Protected movie and castLib files (DXR and CXT) are marginally smaller than their DIR and CST counterparts because they don't include cast member thumbnails or human readable scripts (i.e., the *scriptText of member*). Protected files cannot be opened in Director and are intended to remain external to a Projector. Protected files are compacted (as is done to standard files using `File➤Save and Compact`), but assets are not compressed.

Compressed (Shockwave or "Shocked") format

Compressing a file compresses the assets for Shockwave or local playback and protects the assets (as with protected movies). You should manually compact the file before compressing it. Compressed movie and castLib files

(DCR and CCT) are measurably smaller than their standard or protected counterparts, but compressed files must be decompressed as they are loaded into RAM. This trade-off yields better streaming performance over the Internet, where download time is at a premium. Director 6 and 7 include the ability to use DCR and CCT files wherever DXR or CXT files are allowed (even with a local Projector) but the space saved may not justify the slower load time when using local files on a CD-ROM.

In D6 and D7, internal sounds are compressed as SWA if *Compression* is enabled under `Xtras ➤ Shockwave for Audio Settings`. In D7, JPEG and GIF images imported with the *Include Original Data for Editing* option will be retain their JPEG and GIF compression when shocked.

Note that you should generally use Director movie and castLib files of the same genre; protected movies (DXR files) should use protected castLibs (CXT files), and compressed movies (DCR files) should used compressed castLibs (CCT files).

Table 4-3 lists commands that save movies and castLibs in various formats. Saving the main movie saves both its internal castLibs and the Score. Some commands also save the external castLibs linked to the main movie. None of the commands saves MIAWs, which must be opened as the main movie to be edited.

Table 4-3: Saving and Converting CastLibs and Movies

Command	File Type	Compact	Compress	Protect	Replace Original	Batch
File ➤ Save[1,2]	DIR or CST				✓	
File ➤ Save As[3]	DIR or CST	✓			Optional	
File ➤ Save and Compact[2]	DIR or CST	✓			✓	
File ➤ Save As Shockwave movie	DCR or CST		✓	✓		
File ➤ Save as Java[4]	DJR	✓		✓		
File ➤ Save All[1,5]	DIR or CST				✓	
Xtras ➤ Update Movies ➤ Update[6]	DIR or CST	✓				✓
Xtras ➤ Update Movies ➤ Protect	DXR or CXT	✓		✓		✓
Xtras ➤ Update Movies ➤ Convert to Shockwave movie(s)	DCR or CCT		✓	✓		✓

[1] Performs "incremental" save. Option is active only if changes have been made, and only those movies or castLibs that have changed are saved.
[2] This command saves the "active entity" and its components. If the active window is associated with the main movie or any of its externally linked casts, all of these components are saved if necessary. To save an unlinked external castLib, make sure it is the active window.
[3] If the main movie is active, File ➤ Save As saves only the movie, *not* its external castLibs. If a linked or unlinked external castLib is active, it saves only the active castLib and not the main movie.
[4] Creates a Java (DJR) file and optional Java source and class files. Requires Java Export Xtra included with D6.5 and D7.
[5] Saves any open movies or castLibs that have been modified.
[6] Batch updates movies and castLibs from D4 or D5 to D6, or from D5 or D6 to D7.

CastLib preferences

The `File` ➤ `Preferences` ➤ `Cast` option, except for the Label option, can be set separately for each Cast window. They are not castLib properties and cannot be accessed via Lingo.

Maximum Visible

Controls the number of thumbnails (from 512 to 32,000) visible in the Cast window. Set it slightly larger than *the number of members of castLib* to afford finer control with the vertical scrollbar in the Cast window and to improve performance marginally. If you set it too low, you won't see all available cast members, but can still access them via the media editor windows.

Row Width

Controls how many thumbnails are displayed across the Cast window. The fixed options (8, 10, and 20) prevent the cast members from wrapping as the Cast window size changes. When using a fixed number of thumbnails per row, if the Cast window is too narrow, cast members will seem to be missing. Use *Fit to Window* to wrap the display to the window's width.

Thumbnail Size

Use a smaller thumbnail size to see more cast members.

Label

Controls whether the cast member number, name, or both are shown in the Cast window, Sprite Toolbar, Sprite Inspector, and in the Score when the *Display* mode is *Member*.

Media Type Icons

Set this to *All Types* to display the small icons shown in Figure 4-3 to identify each asset type within its cast member thumbnail in the Cast window, Sprite Toolbar, and Sprite Inspector.

Show Cast Member Script Icons

Select this option to distinguish cast members with cast member scripts attached. A small icon (separate from the media type icon) appears at the left of the thumbnails in the Cast window, Sprite Toolbar, and Sprite Inspector. To visually distinguish sprites with attached cast member scripts, use the Director 5 Style Score with the *Behavior* display mode (cast scripts are indicated by "+" signs if no sprite script is attached).

Importing, Inserting, and Creating Assets

You will often create your assets in some external program and then import them into Director. You can also create bitmaps, text, and buttons in Director. Shockwave audio can be exported from SoundEdit or Peak LE on the Mac, or created using the `Xtras` ➤ `Convert WAV to SWA` option under Windows.

Director requires the MIX (Media Information Exchange) Xtras to import various sound and bitmap formats. Without the MIX Xtras (in the *MIX* subfolder of the *Xtras* folder), the corresponding file types will not appear in the File Import dialog box or work via drag-and-drop.

Importing Media into the Cast

Director for Macintosh will import files with either a recognized file extension or the corresponding Macintosh File Type shown in parentheses in Table 4-4. Macintosh File Types are always four characters, case-sensitive, and space-sensitive (the spaces in "BMP " and "RTF " are required). Director for Windows files imports files based only on their three- or four-letter extension. Name all your files with no more than eight characters followed by a three-letter extension. It will make life easier, when copying files across networks with some Windows systems.

Refer to the TechNote, "File Types, Creator Codes and Extensions" at *http://www.zeusprod.com/technote/filetype.html* and Chapter 14 in *Lingo in a Nutshell* for more details on Macintosh File Types and cross-platform file names.

A database of Macintosh Creator Codes and File Types is available from:

http://www.angelfire.com/il/szekely/index.html

QuickTime Pro reads and writes numerous file formats:

http://quicktime.apple.com

DeBabelizer by Equilibrium Technologies reads and writes numerous file formats:

http://www.equilibrium.com/ProductInfo/DB3/DB3ReadersWriters.html
http://www.equilibrium.com/ProductInfo/DBPro/ProReadersWriters.html

IrfanView32 is a freeware graphics file viewer for Windows 95/98/NT:

http://members.home.com/rsimmons/irfanview/

The Shockwave 6.0 plug-in supported any linked bitmap and sound types for which MIX Xtras were installed. The Shockwave 6.0.1 plug-in recognizes GIF, JPEG, and audio files without any Xtras, but ignores any installed MIX Xtras.

Table 4-4 shows the file formats that can be imported using `File` ➤ `Import` or via drag-and-drop. Drag-and-drop import uses the default import settings. For example, you can't use drag-and-drop to create a linked sound, because the default is to create an unlinked sound. Refer to Macromedia's *Using Director* manual for additional details on importing, and see "Import options: To link or not to link" later in this chapter. See also Table 4-5 for additional media types not imported via the `File` ➤ `Import` menu option and requiring Sprite asset Xtras as described in Chapter 10. The Director 6 and 7 CDs contain sample graphics and audio files under *Macromedia/Support*, with which you can practice importing each media type.

Table 4-4: Supported Import File Formats

Asset Type	Notes	Mac	Win
All Files	Shows all asset types	Shows only recognized file types	Imports unknown types as OLE
Bitmap Image[1,2]	All supported graphical types (including PICTs; JPEG import requires QuickTime)	.BMP ('BMP '), .GIF ('GIFf'), .JPG, .JPEG ('JPEG'), .LRG, .PCT, .PIC, .PICT ('PICT'), .PNT ('PNTG'), .PSD ('8BPS'), .TGA ('TPIC'), .TIF ('TIFF')	.BMP, .DIB, .EPS , .FCC, .FCI, .GIF, .JPG, .LRG, .PCD, .PCT, .PCX, .PIC, .PICT, .PNG, .PNT, .PSD, .TIF, .TGA, .WMF
PICT	PICTs only	.PCT, .PIC, .PICT ('PICT')	.PCT, .PIC, .PICT
Palette[1,2]	Create palette in deBabelizer or Photoshop	.PAL ('8BCT'), imported with bitmap[2]	.PAL, imported with bitmap.[2] Photoshop CLUT
Scrapbook	Mac only	'scbk'	N/A
PICS	Mac only	.PICS ('PICS')	N/A
FLC and FLI	Win only (AutoCAD)	N/A	.FLC, .FLI
Sound[3]	Supports uncompressed and IMA-compressed sounds	.AIF, .AIFF ('AIFF'), .AIFC ('AIFC'), .WAV, .WAVE ('WAVE'), System 7 SND ('sfil'), .au ('ULAW'), .SWA ('SWaT'), .MP3 ('MPG3')	.AIF, .AIFF, .AIFC, .WAV, .WAVE, .au, .SWA, .MP3
Director movie[4]	Director 5, 6, or 7 movie files	.DIR ('MV07', 'MV97', 'MV95'), .DCR ('FGDM'), .DXR ('M!07', 'M!97', 'M!95')	.DIR, .DCR, .DXR
Director Cast[5]	Director 5, 6, or 7 cast files	.CST ('MC07', 'MC97', 'MC95')	.CST
Digital Video	Mac: QuickTime Win: Video Clip	.MOV ('MooV'), .MPG, .MPEG ('MPEG'), .AVI ('VFW ')	.MOV, .AVI, .MPG, .MPEG
Rich Text (D6)[1,6] Text (D7)[1]	MS Word creates RTF files	.RTF ('RTF '), .TXT ('TEXT'), .HTM, .HTML	.RTF, .TXT, .HTM, .HTML
Animated GIF	New in D7	.GIF ('GIFf')	.GIF
Shockwave Flash	New in D7; use Insert menu in D6.5	.SWF ('SWFL')	.SWF

[1] This asset type can also be created in Director on either platform.
[2] When importing bitmaps containing custom palettes, Director optionally imports the palette as well.
[3] Director 6 doesn't import SWA via File ➤ Import; D7 does, but they are converted to standard internal sounds. The Sun AU Import Xtra included with D6.5 and D7 is required to import .au sound files. Some compressed WAVE files are not supported.
[4] Importing a Director Movie file directly into the Cast (unlinked) imports all its assets as separate cast members. Its Score becomes a film loop, and its scripts, bitmaps, sounds, and so on are each transferred as separate cast members.
[5] Importing a Director Cast file does not create a cast member. It simply links an existing castLib into the current movie. See Modify ➤ Movie ➤ Casts. ImportFileInto can link to cast files at runtime.
[6] A new rich text cast member is created whenever a page break or column break is encountered. HTML files are imported as rich text cast members, but none of the HTML tags are obeyed in D6. D7 imports HTML files as text members and supports basic tags.

File import notes

All bitmaps are imported at 72 dpi (dots per inch). A 300 dpi bitmap will appear about four times larger upon import. Director flattens a Photoshop document's multiple paint layers rather than importing then as separate elements. Either export the layers as separate images from Photoshop, or use the Photocaster Xtra (*http://www.medialab.com)* to import the Photoshop layers as separate cast members automatically (Photocaster Lite is included with D7).

The following file types use the file extensions shown in parentheses: xRes (.LRG), Photoshop 3.0 (.PSD), MacPaint (.PNT), and TARGA (.TGA). The Windows-only Postscript (.EPS), Photo CD (.PCD), Windows Meta-File (.WMF), and .PCX formats are imported with the ImageMark MIX Xtra. It imports only the TIFF preview available in some EPS files and not true EPS data. The ImageMark Xtra is not included with D7, so the EPS, PCD, WMF, and PCX formats are no longer supported. The ImageMark MIX Xtra is not licensed for redistribution (i.e., it is for authoring only).

In Director 5, all PICT files were imported at 32-bit, but Director 6 removes this limitation. Under Windows, 16-bit PICT files will import at the current color depth or as 24-bit PICTs. Set your monitor to 16-bit color (thousands) to import PICTs at 16-bit. QuickTime is required to view JPEG-compressed PICT files. In Director 6 and 7, under Windows, PICT files should use a .PCT extension, whereas in Director 5, the .JPG extension was required for JPEG-compressed PICTs.

The Photoshop CLUT palette file import was briefly released with D6.0.2 and reintroduced in D7. Custom palettes are typically imported along with the bitmap file in which they are embedded. In D7, GIFs imported via `File ➤ Import` can be imported as either bitmap or animated GIF members.

Additional media types

Table 4-5 shows additional supported media types that are not imported via the `File ➤ Import` menu option and don't support drag-and-drop. Data types used by sprite Xtras must be inserted using the `Insert` menu or `Xtras` menu in D6.5, although some are imported via `File ➤ Import` in D7. To import Freehand files, you must convert them to Flash format. Note that the .MOV extension is used for both QTVR and linear QuickTime movies, although the two are quite different.

Table 4-5: Additional Supported Formats

Asset Type	To Add Element	Mac	Win
ActiveX[1,2]	Insert ➤ Control ➤ ActiveX	N/A	.OCX
Shockwave Audio[3,4,5]	Insert ➤ Media Element ➤ Shockwave Audio	.SWA ('SwaT')	.SWA
Other Sound formats	Copy Sound Edit 16 and scrapbook sounds to clipboard and paste into the Cast window	N/A	N/A
Custom Cursor[1]	Insert ➤ Media Element ➤ Cursor	None	None
OLE[6]	Insert ➤ Media Element ➤ OLE Object	N/A	Various

Table 4-5: Additional Supported Formats (continued)

Asset Type	To Add Element	Mac	Win
QTVR 1.0	Don't import. See Chapter 16.	.MOV ('MooV')	.MOV
QTVR 2.0[1]	Insert ➤ Media Element ➤ QuickTime 3	.MOV ('MooV')	.MOV
QD3D	See Chapter 16.	'3DMF'	.QD3D
QuickTime 3[1,5,7]	Insert ➤ Media Element ➤ QuickTime 3	.MOV ('MooV'), .AVI ('VfW ')	.MOV, .AVI
Push button, check box, radio button	Use Tool Palette or Insert ➤ Control	Cmd-7	Ctrl-7
Fields	Cut and paste text into fields, use FileIO, or Insert ➤ Control ➤ Field	Cmd-8	Ctrl-8
Flash[1,5,8]	Insert ➤ Media Element ➤ Shockwave Flash movie (D6.5). Insert ➤ Media Element ➤ Flash Movie (D7).	.SWF ('SWFL')	.SWF
PowerPoint[1,9]	Xtras ➤ Import PowerPoint File	.PPT	.PPT

[1] Requires an Xtra included with D6.5 and D7.
[2] The ActiveX Xtra is for Windows only.
[3] Requires an Xtra included with D6.0.x, D6.5, and D7.
[4] SWA can be created using Sound Edit 16 on the Macintosh or Xtras ➤ Convert WAVE to SWA under Windows.
[5] Also imported via File ➤ Import in D7.
[6] OLE is for Windows only. Any file extensions not recognized by Director For Windows are imported as OLE cast members. See Edit ➤ Paste Special ➤ Using OLE. OLE objects created under Windows appear as bitmaps on the Macintosh.
[7] Quicktime 3 supports dozens of media formats, although some of these are usually imported directly. For example, GIF and JPEG files should be imported as bitmaps instead of Quicktime 3 cast members. In D7, import QT3 cast members via File ➤ Import. See Chapter 16.
[8] D6.5 supports Flash 2. D7 supports Flash 2 and Flash 3.
[9] Only PowerPoint 4.0 files are supported. Save PowerPoint 97 files in PowerPoint 4.0 format before importing.

Linked and Unlinked Media Types

Table 4-6 shows each asset type as returned by *the type of member* property, and whether it is linked (asset file remains external) or unlinked (data is embedded into the Cast and stored in Director's native format). Linked assets must be distributed with your Projector.

Some asset types, such as *#digitalVideo*, have additional subtypes reported by a second Lingo property.

Table 4-6: Media Types and Subtypes

Media Type	Notes	Linked?
#ActiveX[1]	Requires ActiveX control included with D6.5 and D7 (Windows only).	Yes
#alpha[1]	Requires Alphamania Xtra (*http://www.medialab.com*).	No

Table 4-6: Media Types and Subtypes (continued)

Media Type	Notes	Linked?
#animGIF[1]	New In D7. See Insert ➤ Media Element ➤ Animated GIF, and File ➤ Import.	Optional
#bitmap	Only unlinked images can be edited in Paint window.	Optional
#btned[1]	Requires Custom Button Editor Xtra. Source bitmaps used for custom button can be deleted. Obsolete in D7.	No
#button	See *buttonType of member* (#checkBox, #pushButton, #radioButton).	No
#cursor[1]	Requires Custom Cursor Xtra included with D6.5 and D7.	No
#digitalVideo	QT2.x or AVI files in D6. In D7, used only for AVI files under Windows. See *digitalVideoType of member* (#quickTime or #videoForWindows).	Yes[2]
#empty	Unoccupied cast member.	N/A
#field	See *boxType of member* (#adjust, #fixed, #limit, #scroll).	No
#filmloop	Cast members used by film loops must be retained.	No
#flash[1]	Requires Flash Asset Xtra included with D6.5 and D7.	Optional
#font[1]	New in D7. See Insert ➤ Media Element ➤ Font.	No
#movie	Only linked Director movies become #movie cast members. If imported as unlinked, the components are imported as different types.	Yes
#ole	Windows-only; treated as #bitmap on Macintosh.	Yes
#palette	See *palette of member* and *paletteRef of member* in Table 13-8.	No
#picture	Use *Import PICT file as PICT* option. Can't be edited in Paint window.	No
#PopMenu[1]	Requires PopUp Xtra (*http://www.updatestage.com/xtras*).	N/A
#QD3D_Xtra[1,3]	Requires QD3D Xtra (included on D6 CD and free from Macromedia).	Optional
#QuickTimeMedia[1,2,3]	Requires QuickTime 3 Asset Xtra included with D6.5 and D7.	Yes
#richtext	To find rich text members, search for members of type *Text* under Edit ➤ Find ➤ Cast Member in D6.5. Obsolete in D7. See #text.	No
#script	See *scriptType of member* (#score, #movie, #parent).	No
#shape	See *shapeType of member* (#line, #oval, #rect, #roundRect).	No
#sound	AIFF and WAVE sound files can be playing using *sound playFile* without a cast member reference. See also #SWA.	Optional
#SWA[1]	Requires SWA Xtras.	Yes
#text[1]	New in D7. Replaces #richText and supersedes #field cast members. (#text type also referred to #field cast members in D4.)	No

Table 4-6: Media Types and Subtypes (continued)

Media Type	Notes	Linked?
#transition	See *transitionType of member* in Table 16-1 in *Lingo in a Nutshell*.	No
#vectorShape[1]	New in D7. See Window ➤ VectorShape.	No
#xtra[1]	Xtras generally report a custom type name. If the required Xtra is missing, sprite may appear as a red X on the Stage.	Xtra-dependent

[1] To find this cast member type (and all Xtra cast members) search for members of type Xtra under Edit ➤ Find ➤ Cast Member in D6. In D7, Vector Shape, QuickTime 3, Flash, Animated GIF, Cursor, Font, and SWA cast members can be searched for individually.
[2] Digital Video cast members are always linked. Don't import QTVR movies as #digitalVideo cast members. Use the QTVR Xtra or the QT3 Xtra instead. QuickTime 2 videos can be imported as #digitalVideo or inserted as #QuickTimeMedia in D6.5. QuickTime 3 videos should be inserted as #QuickTimeMedia.
[3] QD3D cast members can be inserted as #QD3D_Xtra cast members if the QD3D Xtra is installed. They can also be inserted as #QuickTimeMedia if the QT3 Xtra is present.

The *new* command creates cast members on the fly. See Example 3-7. Figure 4-3 shows the icons for most of the media types in Table 4-6. Note that the larger rectangular icons indicate linked assets. Note the new icon for Behavior scripts in D6 and D7.

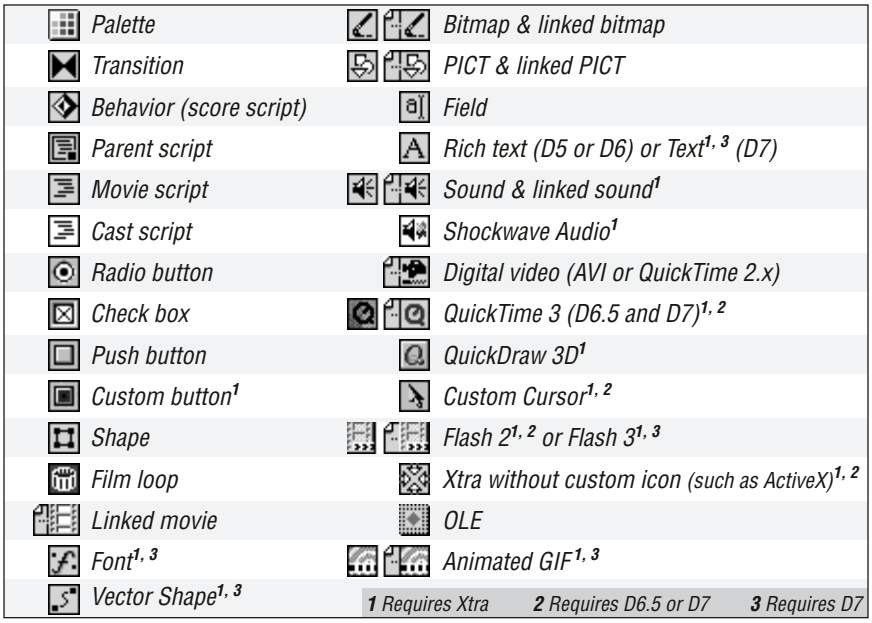


Figure 4-3: Media type icons

Import options: To link or not to link

Refer to the *Import Command* entry in the online Help for details on the basic use of the Import dialog box. The *Internet* button lets Director import files (which can

end up either linked or unlinked) from a URL. When you import a bitmap with a custom palette or with a different color depth than the current movie, Director will prompt you with additional import options. See Chapter 13 for details.

There are four possible import modes:

Standard Import
: Imports media directly into the Cast. Regardless of the external file's format, bitmaps and sounds that are imported in this mode are converted to Director's internal data formats. Once imported, you can't distinguish between a TIFF, PICT, BMP, and so on. If importing a Director movie file, this causes the imported movie's assets to be copied as individual cast members into the main movie's Cast.

Link to External File
: Assets remain in external file(s) and are pointed to by the *fileName of member*, *URL of member*, or *streamName of member* property (see Example 4-5 and the *linked of member* property). This is relevant for AIFF, WAVE, bitmap, PICT, animated GIF, Flash, and Director movie cast member types. Linked filenames are updated automatically for the current platform (Director changes the drive letter and path separators as long as the file's position relative to the Director movie is maintained). Stick with DOS-style "eight dot three" filenames for maximum compatibility. The *fileName of member* updates automatically for assets imported via the `Insert` menu in D7, but not in D6.5. See the "Can't find QuickTime 3, SWA, or Flash files at runtime" entry under "Common Importing and Linked File Problems."

Import PICT File as PICT
: Retains the original PICT (shape-based) data from a PICT file. Otherwise, Director converts the imported PICT into a bitmap cast member. See `Edit ➤ Paste Special ➤ As PICT` (Mac only).

Include Original Data for Editing
: Director retains the original external file's format information, allowing it to be edited in an external editor specified under `File ➤ Preferences ➤ Editors`. In D7, use this option to allow internal GIF and JPEG cast members to be compressed for Shockwave delivery. External editors can be set for AIFF, AVI, BMP, EPS, GIF, JPEG, MacPaint, PAL, PCD, PCX, Photoshop 3.0, PICT, PNG, QuickTime, System 7 *snd* resource, Sun AU, TARGA, TIFF, WAVE, WMF, and xRes LRG file formats.

The import mode is ignored for some file types. For example, digital video assets are always linked and text cast members are always embedded.

Advantages to linking:

- Easy to swap external assets without editing Director movie or cast.
- Importing does not consume a lot of memory.
- Size of castLib is minimized.
- Audio streams from disk at runtime, using less memory.

Disadvantages to linking:

- External unprotected assets must be included with the Projector.
- Linked graphics load more slowly because they are not stored in Director's native file format.
- Linked audio streaming from disk may interfere with loading of other assets.
- Linked sounds don't obey *the loop of member* setting.
- Linked sounds are immediately purged from memory.
- Requires MIX import Xtra(s) at runtime.
- Palettes not always handled properly (see Chapter 13).

Advantages to importing directly into the Cast:

- Movie's assets are contained within Cast or Projector, offering some security, and not requiring external files to be included with the Projector.
- Uses fewer external file handles, although this is rarely an issue.
- Internal audio remains in memory and can be looped.
- Assets are stored in order in which they are used in Score.
- Assets are stored in Director's native format for faster loading.
- Does not require MIX import Xtra(s) at runtime.

Disadvantages to importing directly into the Cast:

- Cast can grow very large.
- Memory can run low when importing.
- Memory can run low when playing large sounds.
- Large sounds must be loaded in their entirety before playing.

Import short or frequently used sounds into the Cast. Leave longer sound files on disk and link to them instead. Before playing an internal sound, Director loads the entire sound into memory. Using linked cast members or *sound playFile* streams the data from disk as it is needed. While it uses less memory, streaming from disk may interfere with the loading of other assets.

Importing tips, annoyances, and caveats

Here are some tips on importing efficiently:

- Drag and drop to import files from the desktop.
- Files are imported in the order in which they are provided by the operating system, not the order in which they are added in the *Import* dialog box, nor necessarily in alphabetical order. Select the newly imported cast members and choose `Modify ➤ Sort` to rearrange them by their name or another attribute.
- Use the Macintosh shareware *Default Folder* extension to make it easier to import media from a variety of subfolders without manually navigating among them.

- When you select items to import in the *Import* dialog box, Director always jumps to the beginning of the available file list. For easier importing, move all the files to be imported into a separate folder, then use the *Add All* button.
- If importing most but not all files in a folder, first use *Add All*, then remove the ones that you don't want to import.
- Imported assets may be scattered around the Cast. Select an empty area in the Cast with enough room to import all assets together before importing.
- If you run out of memory while importing, you should allocate more RAM or import in multiple steps and save between imports. Separate the files into temporary folders and import one folder at a time.

Watch out for these issues:

- Director 6 and 7 use the name of the imported file to name the new cast member but strip off the file extension (unlike D5). If you import two files with the same filename but different extensions, they will be given identical cast member names. See "Checking for duplicate cast member names" later in this chapter. You could write you own utility to add an appropriate extension to each cast member, such as a .BMP extension to bitmap cast members.
- On the Macintosh, you can preview sounds and graphics in the `File ➤ Import` dialog box, but only before adding them to the import list.
- Some cast member types are always linked or embedded regardless of the mode chosen for importing the files. See Table 4-6.

Common importing and linked file problems

Some common problems with importing and linked files are:

Where is . . . ? (File Can't Be Found)

If you link to an external asset file, Director will look for that file when the cast member is needed. You must distribute any external asset files with your Projector. Director automatically converts the linked file path to a path that is relative to the Director movie. It also adjusts the file path from Macintosh to Windows (or vice versa). You should obey Windows 3.1's more restrictive file-naming conventions on any other OS if you intend to distribute under Windows 3.1. Refer to the TechNote, "Path and File Specifications" at *http://www.zeusprod.com/technote/filepath.html*.

If Director can't find a file, it will bring up the dreaded *Where is . . . ?* dialog box. Simply point to the new location for the asset, and Director will update *the fileName of member* property accordingly when you save the file. If you don't have the asset available and don't want to change the link, hit *Cancel* and Director will prompt you again next time you use the file. Test linked files from within your Projector. If a link is incorrect, you will be prompted every time the Projector runs, because Director doesn't save changes automatically from a Projector.

The simplest way to avoid the problem is to keep external files in the same folder with the Director movie or Projector. In any case, always keep files in the same relative positions during development and runtime. Refer to the *checkLinks()* utility in Example 4-5.

If Director can't find a movie file needed for a *go to movie* or *play movie* command, you will also be prompted to find the movie. However, Director will not update your Lingo code, even during authoring, and you will get the same error message until you update your Lingo code manually.

Can't find QuickTime 3, SWA, or Flash files at runtime

When inserting assets via the `Insert` menu (QuickTime 3, SWA, Flash, and others), Director 6.x does not automatically create a relative path. Thus, when the assets are moved or burned onto a CD, Director won't be able to find them. Replace the absolute path in the cast member properties dialog box with a relative path using the @ operator to represent the folder in which the Director file resides (it does *not* represent the castLib's folder, which may differ). For example, if a QT3 movie *myVideo.mov* is in a subfolder named *Video*, edit the file name to read *@/video/myVideo.mov* (without quotes) or set its *fileName of member* property to `"@/video/myVideo.mov"` (with the quotes). D7 handles this automatically; there is no need to use the @ operator.

Running out of memory

Assets imported into Director are stored temporarily in memory. If you import many large items, Director will use up all of its available memory. Import fewer items, then save your Director file to free up memory for additional importing. Avoid using *importFileInto* at runtime, as it consumes memory. Import bitmaps at a lower color depth or link to external assets instead of importing them into the Cast, and avoid importing large rich text files. Also allocate more memory to Director.

Bitmaps registration points

Director ignores the registration points set in other programs, such as Photoshop. Use Photocaster (*http://www.medialab.com*) to maintain registration when importing Photoshop documents, or use `Edit ➤ Launch External Editor`, which retains regPoint information (see `File ➤ Preferences ➤ Editors`). The regPoint may display incorrectly when changing *the fileName of member* property. You may need to set the *regPoint of member* and then force Director to recognize it by setting *the picture of member* property to itself:

```
set the regPoint of member whichMember = point (x, y)
set the picture of member whichMember = ¬
    the picture of member whichMember
```

Imported bitmap is wrong size in Paint window

The bitmap was saved at the wrong resolution, such as 96 dpi or 300 dpi. All bitmaps should be saved at 72 dpi before being imported into Director.

Custom palette not imported

Save the bitmap in indexed color mode with an adaptive palette in Photoshop, deBabelizer, or similar graphics program. Director will detect and optionally import the palette along with the bitmap. Custom palettes embedded in QuickTime movies are not recognized. Attach such palettes to a dummy bitmap and import that instead.

Director 6.5 for Macintosh includes a new PICT Import Export Xtra that prevents Director 6.5 from recognizing the custom palette in a PICT file during import. It should be removed from the *Xtras:MIX* subfolder (don't forget to restart Director). Reinstall it only when using the *Save as Java* function.

Creating Media Within Director

Table 4-7 shows the shortcut commands used to create assets within a Director movie.

Table 4-7: Creating and Inserting Media Within Director

Action	**Command**	**Mac**	**Win**
Import Cast Members	File ➤ Import, drag and drop into Cast window, context-sensitive pop-up in Cast window, or Toolbar button	Cmd-R	Ctrl-R
Export frame(s)	File ➤ Export	Cmd-Shift-R	Ctrl-Shift-R
Add bitmap[1]	Insert ➤ Media Element ➤ Bitmap, or Window ➤ Paint	Cmd-5, then hit + button	Ctrl-5, then hit + button
Add rich text[1] (D6) Add text (D7)	Insert ➤ Media Element ➤ Text, or Window ➤ Text	Cmd-6, then hit + button	Ctrl-6, then hit + button
Add palette	Insert ➤ Media Element ➤ Palette	Cmd-Opt-7	Ctrl-Alt-7
Add vector shape	Insert ➤ Media Element ➤ Vector Shape, or Window ➤ Vector Shape	Cmd-Shift-V	Ctrl-Shift-V
Record sound[1,2]	Insert ➤ Media Element ➤ Sound	None	N/A
Add push button, radio button, or checkbox	Insert ➤ Control..., or use Window ➤ Tool Palette	Cmd-7	Ctrl-7
Add field[1]	Insert ➤ Control ➤ Field, or use Window ➤ Tool Palette	Cmd-8	Ctrl-8
Add custom button[3]	Insert ➤ Control ➤ Custom Button	None	None
Add film loop	Insert ➤ Film Loop or copy sprite(s) and paste into cast member slot	None	None

[1] These asset types can be copied from other applications and pasted into Director via the clipboard. Some information, such as rich text formatting, may be lost in the transfer.
[2] Only the Macintosh version of Director supports recording sounds. Under Windows, you'll need an Xtra, such as Focus 3 SoundFX Xtra (*http://www.focus3.com*) or Sound Xtra (*http://www.updatestage.com/xtras*).
[3] Requires Custom Button Editor Xtra. (Obsolete in D7.)

Exporting

Director exports the Stage area only. Reduce the Stage size to the desired output size before exporting. (Prior to D7, the Stage width is limited to multiples of 16 pixels.) If the export fails, make sure that the visible area of the Stage is not blank. Director exports the data in the Score only—any puppeted sprites are ignored. Director does not export individual cast members, but you can copy sounds, text, and bitmaps to the clipboard and then paste them into an appropriate program or place sprites on the Stage to export them. Many Xtras, such as the ScrnXtra (*http://www.littleplanet.com/kent/kent.html*) will capture the screen and export it to a file.

Director 6 for Macintosh exports in PICT, PICS, Scrapbook, and QuickTime 2 formats. Director 6 for Windows can export a DIB file sequence (BMP), or in Video for Windows (AVI) format. The D7 QT3 Export Xtra supports QT3 export on both Macintosh and Windows.

When exporting in QuickTime or Video for Windows format, transitions are not included and each sound may be exported as a separate audio track. Use Adobe Premiere or similar tool to add visual transitions and SoundEdit 16 or similar tool to remix the audio tracks.

Working with Cast Members

If you replace a cast member, all sprites that reference it will use the new asset. This can be great if you want to replace a button on every screen, but troublesome if you meant to replace only some occurrences.

Cast Member Loading

There are three possible settings for castLib loading under `Modify➤Cast Properties`. These control the overall loading of a castLib's assets:

When Needed
: This is the default mode; loads cast members on demand prior to drawing the frame in which they are needed.

Before Frame One
: This mode loads as many cast members as possible in the order in which they are needed in the Score. This increases the initial load delay, but to the extent that memory is available, animations will perform more quickly.

After Frame One
: This mode behaves the same as *Before Frame One*, except that it displays the first frame as quickly as possible before proceeding to load more data.

Refer to *the purgePriority of member* property and Chapter 9, *Memory and Performance*, for details on loading and unloading individual cast members.

Dynamic Linking to Cast Members at Runtime

If at all possible, import all assets ahead of time during authoring. Avoid importing assets at runtime, as it consumes excessive amount of memory. Use *importFileInto* during authoring only. To link dynamically to an external sound, digital video, or bitmap member, set *the fileName of member* property.

If you attempt to set *the fileName of member* property to an invalid file, the property won't update. Check *the fileName of member* after setting it to determine if the relinking succeeded. Even if there is insufficient RAM to read the external file, *the fileName of member* will update. Check that the *picture of member* property is nonzero to confirm that the import succeeded.

Setting *the fileName of member* works best when replacing a cast member with an external file of the same type. Create a dummy cast member ahead of time for each data type that you intend to import. If necessary, create a dummy cast

member on the fly, using the *new()* function. For example, assuming that the PICT file is in the same folder as the Director movie:

```
set dummy = new (#picture)
set the fileName of dummy = the moviePath & "someFile.PCT"
```

You may need to force Director to update the link using:

```
set the fileName of dummy = the fileName of dummy
```

Note that all sprite properties are not updated when setting *the fileName of member* property. This is especially a problem when using, for example, QuickTime movies with differing frame rates. Likewise, palettes for external files are not well-behaved if the palette changes at runtime. Problems with the registration point are common.

Linking to text files is not supported. To read text files on the fly, you can use the FileIO Xtra and assign the result to a text or field member.

Sorting and Searching for Cast Members

To sort cast members, select the ones to be sorted or choose `Edit ➤ Select All` and then `Modify ➤ Sort`. You can sort cast members by the order in which they are used in the Score, their media type, name, or size. Use the *Empty at End* sort option to eliminate any unused cast member slots.

Use `Edit ➤ Find ➤ Selection` to search for a given cast member in the Score. Use `Edit ➤ Find ➤ Cast Member` to locate cast members with a particular name, media type, or palette. You can search all castLibs or limit the search to a single castLib, and list the matching cast members in name or number order. In D6, search for cast members with a Type of *Xtra* to find SWA cast members (searching for *Sound* cast members won't suffice) or to find other Xtra asset types added in D6.5 (QT3, Flash, Custon Cursors, and ActiveX). In D7, you can individually select the new types (Vector Shape, QuickTime 3, Flash, Animated GIF, Cursor, Font, Shockwave Audio) that were lumped together under "Xtras" in D6 and still appear if you search for Xtras in D7. You can use *Select All* to highlight all the found cast members in the Cast window.

Deleting unneeded cast members

Use the *Usage (Not Used in Score)* option under `Edit ➤ Find ➤ Cast Member` to find unused cast members. Movie script cast members are never shown as unused, even though they don't appear in the Score. If a film loop is used in the Score, its constituent members are considered to be used in the Score also. These cast members must be kept handy, as they are not embedded in the film loop. Similarly, cast members included in Custom Cursor members must remain in the cast. Conversely, graphics embedded into Custom Button cast members in D6 may be discarded if not used elsewhere.

Cast members that are not used in the Score may still be used. Parent scripts, fonts, and Custom Cursors are shown as unused in the Score, even though you may well need them. Do not delete them.

So-called unused cast members that are in fact used as *puppetSprites* should be placed in dummy frames of the Score near other related sprites. This prevents them from being flagged as unused and optimizes their storage order on disk for faster loading. Delete truly unneeded cast members and use `File ➤ Save and Compact` to reduce the Director movie's size permanently. Never clear cast members from external castLibs, unless you are sure that other Director movies don't use them either.

Cast Window Shortcuts

The Cast window (Figure 4-1) contains many options that are common to the media editors (Figure 2-1), such as the arrow buttons, the script icon, and the properties icon. It also contains a castLib pop-up menu and a cast member number display.

You can select multiple cast members to check their cumulative size or modify their purge priorities all at once. If you select only bitmap cast members, you can also set their default palette.

To list all the colors used in one or more cast members, select the cast member(s) in the Cast, and then use the Palette window's *Select Used Colors* option.

Table 4-8 lists Cast window shortcuts. Refer to the tables in Chapter 3 for details on creating and manipulating sprites.

Table 4-8: Cast Window Shortcuts

Action	**Command**	**Mac**	**Win**
Cut, copy, paste, or edit cast members	Edit menu, or ontext-sensitive menu	Ctrl-click	Right-click
Modify cast member properties[1]	Modify ➤ Cast Member ➤ Properties (see also Table 2-8)	Cmd-I	Ctrl-I
Modify cast member script	Modify ➤ Cast Member ➤ Script, or use *Script* button.	Cmd-' (apostrophe). Opt-*Script* button opens new script.	Ctrl-' (apostrophe). Alt-*Script* button opens new script.
Edit in appropriate internal media editor	Edit ➤ Edit Cast Member, or select thumbnail and press Return. Double-click thumbnail in Cast, Sprite Inspector, or Sprite Toolbar	None	Alt-E,M

Table 4-8: Cast Window Shortcuts (continued)

Action	Command	Mac	Win
Edit in external editor	Edit ➤ Launch External Editor (see File ➤ Preferences ➤ Editors)	Cmd-, (comma)	Ctrl-, (comma)
Edit cast member name[2]	Click in cast member's name area	Cmd-Shift-N	Ctrl-Shift-N
Switch displayed castLib	Use castLib pop-up menu	Cmd-↑ Cmd-↓	Ctrl-↑ Ctrl-↓
Open additional Cast windows	Window ➤ Cast ➤ *castLib*	Opt-click castLib pop-up	Alt-click castLib pop-up
Jump to specific cast member	Type its number quickly in cast member number field	None	None
Jump to next or previous occupied slot	Click on desired cast member, or use arrow buttons in media editor.	Cmd-→ Cmd-←	Ctrl-→ Ctrl-←
Jump to first cast member	Scroll to top of vertical scrollbar	Home key[3]	Home key[3]
Jump to last used cast member	Use vertical scrollbar	End key[3]	End key[3]
Page up or down one screen in Cast	Click above or below vertical scroll slider	Page Up or Page Down key[3]	Page Up or Page Down key[3]
Select a range of cast members	Select first cast member, then Shift-click last cast member	Shift-click	Shift-click
Select all cast members	Edit ➤ Select All	Cmd-A	Ctrl-A
Create new cast member	Click + button in media editor window	Cmd-Shift-A	Ctrl-Shift-A
Select discontiguous cast members	Edit ➤ Find ➤ Cast Members	Cmd-click	Ctrl-click
Delete single cast member (copies to clipboard)	Edit ➤ Cut Cast Members	Cmd-X	Ctrl-X
Copy cast member(s) to clipboard	Edit ➤ Copy Cast Members	Cmd-C	Ctrl-C
Clear multiple cast members (does not copy to clipboard)	Edit ➤ Clear Cast Members	Delete key	Delete or Backspace key
Duplicate cast member(s)	Edit ➤ Duplicate	Cmd-D, or Opt-drag	Ctrl-D, or Alt-drag
Find or select cast member by name, type, palette, or usage in Score	Edit ➤ Find ➤ Cast Member	Cmd-;	Ctrl-;

Table 4-8: Cast Window Shortcuts (continued)

Action	Command	Mac	Win
Find where used in Score	Select cast member, then Edit ➤ Find ➤ Selection	Cmd-H	Ctrl-H
Exchange cast member used in a sprite[4]	Select sprite, then select cast member, then Edit ➤ Exchange Cast Members, or use Toolbar	Cmd-E	Ctrl-E
Sort cast members by usage in Score, name, size, type	Select at least two cast members then choose Modify ➤ Sort	None	None
Place cast member at center of Stage	Drag *Drag Cast Member* tool in Cast window or media editor to Score	Cmd-Shift-L	Ctrl-Shift-L
Move cast members within Cast window	Select and drag or select, release mouse, and use *Drag Cast Member* tool	None	None
Lay out selected cast members over time	Modify ➤ Cast to Time	Cmd-Shift-Opt-L	Ctrl-Shift-Alt-L

[1] The online Help for "Keyboard Shortcuts" under "Cast window and cast editor shortcuts" is outdated in D6. Ctrl-clicking a thumbnail doesn't edit cast member properties in D5, D6, or D7 as it did in D3 and D4. On the Macintosh, it opens a context-sensitive menu, and under Windows it toggles the selection of discontiguous cast members.
[2] This shortcut was added in D6.0.1.
[3] Use the Home, End, Page Up, and Page Down keys that exist between the numeric keypad and the main keyboard, not the ones on the numeric keypad itself.
[4] Option-double-clicking or Alt-double-clicking does not exchange cast members as claimed in the online Help.

Moving and Copying Cast Members

Director prevents accidental deletion of cast members by disabling `Edit ➤ Cut Cast Members` (`Cmd-X` or `Ctrl-X`) when more than one cast member is selected. Use `Edit ➤ Clear Cast Members` instead.

Deleting cast members that are used in the Score is fraught with peril. Use `File ➤ Find ➤ Cast Members ➤` *Usage* to ensure that the cast members are not used in the Score. Also make sure that they are not used via Lingo.

When you delete a cast member using `Edit ➤ Cut Cast Members`, Director copies the cast member to the clipboard, which can be slow and may overflow memory for large cast members. If you don't need to paste the cast member, use `Edit ➤ Clear Cast Member` or the `Delete` key to delete the cast member instantly. Director prompts you to confirm the deletion only when deleting multiple selected cast members.

Moving cast members in the Cast window

The square *Drag Cast Member* icon (see Figure 4-1) always represents the currently selected cast members. To move the cast members, you need not drag

the selection around the Cast window or wait for it to scroll. Instead, use the following:

1. Highlight one or more cast members and release the mouse.
2. Scroll using the keyboard or Cast window scrollbars.
3. Drag the square icon to the destination. It acts as a proxy for the original selection.

You can also drag the square icon from the Cast window or any of the media editor windows (such as the Paint window) to the Stage or Score.

Whenever you move a cast member, Director updates the Score, but won't update any Lingo code. Refer to cast members by name from within Lingo to avoid problems if they move. Cutting and pasting cast members does not maintain the correct Score references, and should be used with caution or not at all.

Copying cast members between movies

When you copy multiple cast members, a *scrap tag* that identifies the original assets' location is placed on the clipboard instead of the actual cast member data.

When copying cast members between two movies, save the source file first, or the scrap tag may point to the wrong stuff.

You can also copy cast members between movies by using an unlinked external castLib as a conduit:

1. Use `File` ➤ `New` ➤ `Cast` to create an unlinked external castLib.
2. Drag the cast members from the first movie's castLib to the conduit castLib.
3. Close the current movie and open the destination movie.
4. Drag the cast members from the conduit castLib to the second movie's castLib.

When copying sprites or frames, Director also transfers any necessary cast members to the new movie, including linked cast members, which remain linked.

You can replace an entire cast library, and the Score will use the new cast members. This is ideal for simplifying project management (or internationalization), but works only if all the cast members in the replacement cast have the same location as those in the original cast. Otherwise, it wreaks havoc.

Common cast member–related errors

These are some of the most common errors when working with cast members:

Editing a cast member used in multiple places
: Editing a script, bitmap, field, or text cast member that is used in multiple frames or sprites of the Score causes a universal change whether intended or not. Changing the width, height, or text of a field or text cast member or the *bilite of member* property of a button cast member changes them everywhere throughout the Score. Use separate cast members if necessary.

Incorrect Score references

If you move member(s) in the Cast, the Score will update automatically to point to the cast members' new locations, but there are several actions that can lead to incorrect Score references.

Changing a cast member's position via cutting and pasting will not update the Score. You can (carefully) paste a replacement cast member into the old one's position in the Cast window, but if you copy and paste cast members incorrectly, a Score reference might point to the wrong type of asset. For example, the sound channel may point to a bitmap cast member. Errant script references can be created if you cancel a new script, as described in Example 2-2 in *Lingo in a Nutshell*. This can be very confusing and difficult to debug. See Example 3-9 to detect this type of corruption.

If you delete a cast member that is referenced in the Score, Director won't be able to find it. Director will repeatedly try to load the nonexistent cast member, and this may crash Director. Use `Edit ➤ Find ➤ Cast Member ➤ Usage` to make sure a cast member is not used before deleting it.

Corrupted files or cast members errors

Though not common, it is not exceedingly rare for a file or individual cast member to become corrupted. If a file appears corrupt, use `File ➤ Save As` or `File ➤ Save and Compact` to recover it. In severe cases, copy and paste the Score and/or cast members to a new movie. If an individual cast member is corrupted (as indicated by an "Error Unpacking Cast Member" error), replace it with a backup or placeholder. Use `Edit ➤ Find ➤ Selection` to find where it is used in the Score and to remind you of the nature of the lost cast member. Use `Edit ➤ Clear Cast Members` instead of `Edit ➤ Cut Cast Members` to delete corrupted cast members.

Do not use older versions of Norton Utilities on a Mac OS8 HFS+ file partition, as it can corrupt your files.

Memory errors

An "Out of Memory" or "Not Enough Memory To Load This Cast Member" error may indicate that the Score is trying to load a non-existent cast member. If low on memory, use `Edit ➤ Clear Cast Members` instead of `Edit ➤ Cut Cast Members`. The latter attempts to copy the item to the clipboard, which is slower and requires more memory. Save the file frequently to free memory consumed by pending changes.

CastLib and Cast Member Lingo

Most Lingo member-related commands accept a cast member reference of the form:

```
member whichMember {of castLib whichCast}
```

where *whichMember* and *whichCast* can be names or numbers, such as:

```
member "Headline"
member "Headline" of castLib 7
member "Background" of castLib "newArt"
member 1 of castLib 3
```

or, in D7 notation:

```
member("Headline")
member("Headline",7)
member("Background", "newArt")
member(1,3)
```

D7 will not tolerate member references of the form *member (x) of castLib y*. Convert them to *member (x,y)*. See the D7 *ReadMe* file for details.

If the optional castLib is not specified, Director may assume the first (internal) castLib, *the activeCastLib*, or the current castLib of a sprite's associated member depending on the command used, so specify an explicit castLib when in doubt. It is generally a good idea to refer to castLibs by name rather than number in case the order of castLibs changes.

The *erase member* command deletes cast members without a confirmation. The *move member* function does not update the Score notation to reflect the cast member slot changes, which will probably lead to incorrect Score notation.

Access Speed and Name Caching

You can refer to cast members by number, but because cast member numbers may change, you should access members by name. Director always finds the first cast member with the specified name, so you should take care to avoid duplicate cast member names (see Example 4-4).

Prior to Director 5, accessing cast members by name was slow, because Director looked up the cast member each time. As of Version 5, Director caches the names of cast members the first time they are used. Subsequent accesses by cast member name are comparable in speed to access by cast member number. Even so, cast members that appear earlier in the cast are found more quickly the first time when searching by name.

The number of member property is convenient for finding a member by name; it returns –1 if the member is not found:

```
put the number of member "existing member"
-- 5
put the number of member "nonexistent"
-- -1
```

Director does not cache script name references. The following can be very slow:

```
repeat with x = 1 to 100
  set myObj = new (script "Parent Script")
end repeat
```

The following can be significantly faster when creating many script instances:

```
set n = the number of member "Parent Script"
repeat with x = 1 to 100
```

```
    set myObj = new (script n)
  end repeat
```

If you add or delete cast members during authoring, the name cache may become inaccurate. For example, deleting a cast member may not be reflected immediately in *the number of member* property, which should return –1, but instead returns the old member number:

```
put the number of member "deleted member"
-- 7
```

Closing and reopening the file should reset the name cache.

Table 4-9 covers Lingo commands that operate on a castLib or create, move, or delete members within a castLib. See Table 4-10 for a complete list of cast member and sprite properties.

Table 4-9: CastLib and Cast Member Lingo

Lingo	**Usage**
the activeCastLib	Returns the number of the currently selected castLib. Buggy in D7.0, but fixed in D7.0.1.
castLib *whichCast*	Refers to a castLib within an expression, e.g.: `put the name of castLib whichCast`
the castLibNum of member *whichMember*	Returns the number of the castLib containing a particular cast member. (Read only.)
duplicate (member *fromMember* {of castLib *fromCast*} {, member *toMember* of castLib *toCast*})	Duplicates the specified cast member. Returns new cast member position.
erase (member *whichMember* {of castLib *whichCast*})	Deletes the specified cast member (dangerous!). Always returns 0.
the fileName of castLib *whichCast*	Returns the complete path to a castLib file.[1] Can be set for external castLibs.
findEmpty(member *whichMember* {of castLib *whichCast*})	Finds the next available cast member slot in a castLib. If you don't specify a castLib, it assumes castLib 1, not *the activeCastLib*.
importFileInto member *whichMember* {of castLib *whichCast*}, *fileNameOrURL*	Imports an asset into a castLib. Not recommended at runtime, because it consumes memory.
member *whichMember* {of castLib *whichCast*}	Refers to a member within an expression, e.g., `put the name of member 1 of castLib 1`
member (*whichMember*, *whichCast*)	Refers to a member in D7 notation.
move (member *fromMember* {of castLib *fromCast*} {, member *toMember* of castLib *toCast*})	Moves the specified cast member, but does not update Score notation! Existing cast member in destination will be replaced. Returns new cast member position.

Table 4-9: CastLib and Cast Member Lingo (continued)

Lingo	Usage
the movieFileFreeSize	Returns the number of bytes saved by performing a File ➤ Save and Compact to purge deleted members.
the name of castLib *whichCast*	Specifies the name of the castLib. Can also be set.
new(*#memberType*)	Creates a new cast member on the fly. Returns the member reference of the newly created cast member.
the number of castLib *whichCast*	Returns the number of a castLib specified by name.
the number of castLibs	Returns the totals number of castLibs (both internal and external) attached to the movie.
the number of members of castLib *whichCast*	Returns the number of the highest cast member slot used in a castLib even if those cast members have been deleted.
the preLoadMode of castLib *whichCast*	Determines when cast members will be loaded. See Modify ➤ Cast ➤ Properties.
save castLib *whichCast*, *destinationFile*	Stores a castLib to disk. Use it to export an internal castLib. Works with external protected castLibs.
saveMovie *destinationFile*	See Chapter 6, *The Stage and Movies-in-a-Window*.
the selection of castLib *whichCast*	Specifies the cast member(s) highlighted in the specified castLib. Can be tested and set.

[1] The *fileName of castLib* of the first internal castLib is the same as the movie's complete path. The *fileName of castLib* of any secondary internal castLibs is EMPTY. The *fileName of castLib* of an external cast is its complete file path. This property can be set for external castLibs, but any changes are ignored if the specified file does not exist. Setting this property for internal castLibs has no effect.

The utility in Example 4-1 displays all the castLibs and indicates whether they are internal or external.

Example 4-1: Listing Internal and External castLibs

```
on showCastLibTypes
  repeat with i = 1 to the number of castLibs
    case (the fileName of castLib i) of
      EMPTY, the moviePath & the movieName:
        set castLibType = "Internal"
      otherwise
        set castLibType = "External"
    end case
    put "CastLib" && i && the name of castLib i && castLibType
  end repeat
end showCastLibTypes
```

Creating New Cast Members on the Fly

The *new(#memberType)* function (see Example 3-7) returns a cast member reference only if it succeeds. It returns error code –2147219501 if the desired type is not available, perhaps because of a missing Xtra. For example, *new(#flash)* will fail if the Flash Asset Xtra is not available. Beware—if you get *#memberType* as a return value, you most likely have an *on new* handler in a movie script that is intercepting the *new()* function call. Any *on new* handlers should reside in score scripts or parent scripts only.

When you create a new cast member on the fly, you may need to set its properties, such as its *picture of member*, *media of member*, or *text of member*. Note that *new(#script)* creates a movie script. Set *the scriptType of member* to *#score* or *#parent* as needed. Note that *new(#shape)* creates a #rect. Set *the shapeType of member* to *#roundRect*, *#oval*, or *#line* as needed.

Cast Member and Sprite Properties

Cast member and sprite properties are at the heart of Lingo and Director. A single cast member has a single set of member properties, but each instance in which it is used as a sprite can have a unique set of sprite properties. Sprite properties always pertain to the sprites in the current frame. Sprites that have been manually puppeted while in another frame override the current frame's Score notation.

There is no easy way to read or set the properties of sprites in frames other than the current frame. In D6 and D7, it is best to have a sprite change its own properties when that sprite is finally reached. If necessary, store the new sprite properties in global or property variables that can be accessed in the *on beginSprite* handler, which is called before a sprite is drawn.

To check sprite properties in another frame, use something of the form:

```
set oldFrame = the frame
set the updateLock = TRUE
go frame someFrame
if the property of sprite someSprite = someValue then
  statement(s)
end if
go frame oldFrame
```

Understanding cast member and sprite properties

Most cast member and sprite properties can be set via Lingo, and many read-only properties can be set indirectly or via Director's interface. For example, you can change a cast member's width and height using `Modify ➤ Transform Bitmap`, or change the *left*, *top*, *right*, and *bottom of sprite* properties by setting *the rect of sprite* property. D7, unlike D6, allows you to set the *left, right, top,* and *bottom of sprite* directly as well. Some properties are available via Lingo only, such as *the media of member* and *the currentTime of sprite* properties.

It is often possible to guess whether a property pertains to cast members, sprites, or both. Cast member properties tend to be attributes that don't change or are intrinsic to the cast member itself, such as *the sampleRate of member*. Sprite properties often

pertain to a cast member's use on the Stage at a given time, such as *the loc of sprite*. (A cast member does not have a location on the Stage, so a *loc of member* property would make no sense.) Some properties, such as *the width*, are both cast member and sprite properties. A cast member has an intrinsic width, but it can also be resized on-Stage when it is used as a sprite.

All cast members share some properties, but each cast member type may also have unique properties. Likewise, all sprites share some properties, but each sprite type may also have unique properties. Table 4-10 lists the cast member and sprite properties for each asset type. Be sure to test *the type of member* before testing asset-specific properties, such as:

```
if the type of member whichMember = #shape then
  -- We're sure it is a shape, so we can check the shapeType
  if the shapeType of member whichMember = #oval then
    put "We found an oval"
  end if
end if
```

To check if a cast member is empty, use:

```
if the type of member whichMember = #empty then...
```

To check if a sprite is empty, use:

```
if the memberNum of sprite whichSprite = 0 then...
```

or:

```
if the type of sprite whichSprite = 0 then...
```

Lingo Syntax for Cast Member and Sprite Properties

Although not shown explicitly, all cast member and sprite properties shown in Table 4-10 are specified as:

```
the property of member whichMember
the property of member whichMember of castLib whichCast
the property of sprite whichSprite
```

In D7, you can use the equivalent dot notation:

```
member(whichMember).property
member(whichMember, whichCast).property
sprite(whichSprite).property
```

Don't confuse member properties with *the member of sprite* and *memberNum of sprite* properties, which determine a sprite's cast member (such as a bitmap) and can be changed at runtime to change a sprite's appearance.

To refer to a sprite's member, use *the member of sprite* or *memberNum of sprite* property:

```
set the member of sprite (the currentSpriteNum) = ¬
      member whichMember
set the memberNum of sprite (the currentSpriteNum) = ¬
      someMemberNumber
```

In D7, you can use:

```
sprite(the currentSpriteNum).member = member whichMember
sprite(the currentSpriteNum).memberNum = someMemberNumber
```

The *number of member*, *member of sprite*, and *memberNum of sprite* properties all differ. *The member of sprite* uniquely identifies a cast member by both its castLibNum and position (*memberNum*) within that castLib. The *memberNum of sprite* is the integer slot number of a cast member, but does not uniquely identify a cast member, because it doesn't include the castLibNum. The *number of member* property converts a member reference into a unique integer regardless of its castLib.

Adding an integer (*n*) to the *number of member* or *memberNum of sprite* property will indicate a cast member *n* slots away from the original member. Adding an integer to *the member of sprite* doesn't work and results in a zero value.

Note that two sprites may have the same *memberNum*, but actually be two different cast members in two different castLibs. (If using only one castLib, this isn't an issue.)

```
put the member of sprite 11
-- (member 5 of castLib 2)
put the member of sprite 12
-- (member 5 of castLib 1)
put the memberNum of sprite 11
-- 5
put the memberNum of sprite 12
-- 5
```

For backward compatibility the obsolete *castNum of sprite* property returns a unique number identifying the cast member. For members in the first castLib, it is identical to *memberNum of sprite* property. For members in subsequent castLibs, it is equal to:

```
(the castLibNum of sprite) * 65536 + the memberNum of sprite
```

For example:

```
put the member of sprite 11
-- (member 5 of castLib 2)
put the castNum of sprite 11
-- 131077
```

Use *the number of the member of sprite* instead of the obsolete *the castNum of sprite* to obtain this unique number:

```
put the number of the member of sprite 11
-- 131077
```

The *number of member* property reports a different value in movies updated from D4 that used a Shared Cast than it ordinarily does for movies created from scratch in D6. See "Shared Cast versus external cast libraries" earlier in this chapter.

The obsolete *cast of member* and *cast of sprite* properties are not meaningful and should not be used.

A nonexistent member returns the number –1:

```
put the number of member "Nonexistent"
-- -1
```

In D7 notation, *member ("nonexistent").number* generates an error for non existent members.

If a sprite channel is empty, its *member* and *memberNum* properties are as such:

```
put the memberNum of sprite 50
-- 0
put the member of sprite 50
-- (member 0 of castLib 0)
```

You'll often see this *incorrect* attempt to change a sprite's cast member:

```
set the member of sprite 5 = the member of sprite 5 + 1
```

Adding an integer to *the member of sprite* fails because *the member of sprite* is a complex structure. Adding an integer to it performs an implicit type conversion that results in a value of zero!

```
put the member of sprite 5
-- (member 2 of castLib 1)
put the member of sprite 5 + 1
-- 0
```

However, adding an integer to *the memberNum of sprite* works because *the memberNum of sprite* is an integer:

```
put the memberNum of sprite 5
-- 2
put the memberNum of sprite 5 + 1
-- 3
```

Use the following to switch a sprite to display the next cast member in the same castLib:

```
set the memberNum of sprite 5 = the memberNum of sprite 5 + 1
```

The previous example calculates *the memberNum of sprite 5*, and then increments it by one. It does not calculate *the memberNum of sprite 6*.

Use parentheses to refer to a different sprite number, in this case, sprite 6:

```
set the memberNum of sprite 5 = the memberNum of sprite (5 + 1)
```

Set *the member of sprite* instead of *the memberNum of sprite* to switch to a new cast member in a different castLib:

```
set the member of sprite 5 = member 7 of castLib 3
```

The *memberNum of member* property always reflects the offset of the cast member slot from the beginning of its castLib. It doesn't change unless you move the cast member in the Cast window. The *memberNum of sprite* doesn't change unless you set it via Lingo (or edit the Score or move the cast member while the movie is halted).

The *castNum of sprite*, *number of member*, *member of sprite*, and *castLibNum of sprite* properties can vary with the number or order of castLibs attached to a given movie. A single external castLib may have a different castLib number in two movies to which it is attached.

To refer to a sprite itself within a script attached to the sprite use *the currentSpriteNum*, such as:

```
on mouseDown
  set the loc of sprite (the currentSpriteNum) = the clickLoc
end mouseDown
```

You can also use *the spriteNum of me* property (note the required *me* parameter):

```
on mouseDown me
  set the loc of sprite (the spriteNum of me) = the clickLoc
end mouseDown
```

This can also be rewritten as follows (note that *spriteNum* is declared as a property variable):

```
property spriteNum
on mouseDown me
  set the loc of sprite spriteNum = the clickLoc
end mouseDown
```


Some properties, such as *the scale of sprite* and *the duration of member,* use different units when applied to different asset types. Others differ markedly for internal and external assets (*the fileName of member* is `EMPTY` for internal members, but contains the external filename of linked assets; conversely, *the media of member* is meaningful for internal members only). Some properties are stored permanently in the Score; others such as the *rect of sprite* and *the quad of sprite* are secondary properties derived from the Scored properties at runtime.

Setting member and sprite properties

Setting a member property makes a permanent change to the target cast member. Member properties are usually set via *Cast Member Properties* dialog boxes instead of Lingo. The latter is most useful when writing authoring-time utilities. Setting member properties at runtime is allowed, but not necessarily reliable. For example, you cannot reliably change the *directToStage of member* digital video property at runtime. Instead of changing the property at runtime, create two versions of the same cast member with different values for the *directToStage* property and swap a sprite's *member of sprite* as necessary to switch between them.

Auto-puppeted properties and manually puppeted sprites that have been unpuppeted get reset automatically only when a change occurs in the Score. See "Auto-puppets versus manual puppets" in Chapter 1, *How Director Works*, for details.

If possible, set sprite properties instead of member properties at runtime. For example, set a field's *editable of sprite* property rather than its *editable of member* property. Member properties usually update immediately. Sprite properties don't update until the Stage is redrawn using *updateStage* or by the playback head advancing.

Because edits to fields affect the cast member, they appear immediately. (This is a problem in Score Recording even when *the updateLock* is `TRUE`.) When setting member properties at runtime, you can force Director to recognize them by setting the *picture* or *media of member* property to itself, as shown in Example 4-2.

Example 4-2: Setting Member Properties at Runtime

```
set the regPoint of member "myBitmap" = point (50, 38)
set the picture  of member "myBitmap" = ¬
  the picture of member "myBitmap"
set the loop of member "myFilmLoop" = TRUE
set the media of member "myFilmLoop" = ¬
  the media of member "myFilmLoop"
```

By contrast, sprite properties are intended to be both read and set at runtime. Setting a sprite property at runtime via Lingo causes that setting to temporarily override the Score notation. Change a sprite's properties permanently by editing it in the Score or on the Stage when the movie is halted. Lingo changes to sprite properties are stored permanently only if they are made during a Score Recording session.

Table 4-10 is a complete list of member and sprite properties, listed alphabetically by asset type (see the remaining chapters for frame, window, movie, and system properties). The table does not repeat the common properties shared by all cast members (excluding *#empty* ones). Not all member and sprite properties are settable, and all properties are not meaningful for all member types. For example, the *editable* property applies only to text and field assets. Properties such as *rect*, *width*, *height*, and *loc*, apply only to members that have a pictorial representation (bitmaps, shapes, text, video, Flash, etc., but not SWA, transitions, palettes, fonts, or scripts). Obsolete properties (the *cast* and *castType of member* and the *cast*, *castNum* and *immediate of sprite*) are omitted.

Table 4-10: Cast Member and Sprite Properties

Media Type	Cast Member Properties	Sprite Properties
All types (prior to D7)	castLibNum, fileName,[1] height, loaded, media, mediaReady, member, memberNum, modified, name, number, picture, purgePriority, rect, regPoint, scriptText, size, type, width	backColor, blend, bottom,castLibNum, constraint, cursor, foreColor, height, ink, left, loc, locH, locV, member, memberNum, moveableSprite, puppet, scoreColor, scriptInstanceList, scriptNum, rect, right, stretch, top, trails, type, tweened, visible, visibility, width
New properties in D7	thumbnail	bgColor, blendLevel, color, endFrame, flipH, flipV, locZ, quad, rotation, scriptList, skew, startFrame, volume

Table 4-10: Cast Member and Sprite Properties (continued)

Media Type	Cast Member Properties	Sprite Properties
#ActiveX[2]	Each imported ActiveX control has its own custom member properties.	Each imported ActiveX control has its own custom sprite properties.
#animGIF[3]	directToStage, linked, fixedRate, playbackMode	See common properties.
#bitmap[4]	alphaThreshold,[3], depth, dither,[3], palette, paletteRef, picture, useAlpha[3]	See common properties.
#btned[2] (obsolete in D7)	behavesLikeToggle, enabled, initialToggleState, labelString	behavesLikeToggle, enabled, isToggle
#button	alignment, backColor, buttonType, font, fontSize, fontStyle, foreColor, hilite, lineHeight, text	See common properties.
#cursor[2]	automask, cast memberList, cursorSize, hotSpot, interval, type	N/A (see *the cursor of sprite* for other sprite types)
#digitalVideo (used for QT2 in D6, and AVI only in D7)	center, controller, crop, cuePointNames, cuePointTimes, digitalVideoType, directToStage, duration,[5] frameRate, loop, pausedAtStart, preLoad, sound,[6] startTime, stopTime, track, tracks, timeScale, video	currentTime,[7] mostRecentCuePoint,[7] movieRate, movieTime, startTime, stopTime, volume[7]
#empty	memberNum, number, name, type (other common properties not supported)	See common properties (most evaluate to zero for empty sprites).
#field[8]	alignment, autoTab, backColor, border, boxDropShadow, boxType, dropShadow, editable, font, fontSize, fontStyle, foreColor, lineCount, lineHeight, margin, pageHeight, picture,[9] rect, scrollTop, text, wordWrap	editable, rect
#filmloop	center, crop, loop, media, sound, regPoint (readonly)	See common properties.
#flash[2,10]	actionsEnabled, bufferSize, buttonsEnabled, clickMode, eventPassMode, fileName, fixedRate, frameCount, frameRate, linked, loop, pathName, pausedAtStart, percentStreamed, playBackMode, posterFrame, preload, quality, rotation, sound, state, streamMode, streamSize, type, URL Also valid for #vectorShape: broadcastProps, centerRegPoint, defaultRect, defaultRectMode, directToStage, flashRect, imageEnabled, originH, originMode, originPoint, originV, regPoint, scale, scaleMode, static, viewH, viewPoint, viewScale, viewV	bytesStreamed, buttonsEnabled, bytesStreamed, clickMode, directToStage, eventPassMode, fixedRate, frame, loop, mouseOverButton, originH, originMode, pausedAtStart, playBackMode, playing, quality, sound Also valid for #vectorShape: imageEnabled, originPoint, originV, rotation, scale, scaleMode, static, viewH, viewPoint, viewScale, viewV

Table 4-10: Cast Member and Sprite Properties (continued)

Media Type	Cast Member Properties	Sprite Properties
#font[3]	bitmapSizes, characterSet, font, fontStyle, name, originalFont, height, width	N/A (never used as a sprite)
#movie	center, crop, loop, scriptsEnabled, sound	See common properties.
#ole	See common properties.	See common properties.
#palette	See common properties (palette and paletteRef not supported).	N/A (never used as a sprite)
#picture	picture (palette and paletteRef not supported)	See common properties.
#QD3D_xtra[2]	See Table 16-19.	See Table 16-19.
#QuickTime-Media[2,11]	center, controller, crop, cuePointNames, cuePointTimes, directToStage, duration, fileName, frameRate, invertMask, isVRmovie, loop, mask, pausedAtStart, preload, regPoint, rotation, scale, sound, timeScale, translation, type, video	currentTime,[7] duration, isVRmovie, loopBounds, mostrecentCuePoint,[7] mouseLevel, movieRate, movieTime, mRate, mTime, rotation, scale, start-Time, stopTime, timeScale, translation, volume,[7] volumeLevel, VRfieldOfView, VRhotSpotEnterCallback, VRhot-SpotExitCallback, VRmotionQuality, VRmovedCallback, VRnode, VRnodeEn-terCallback, VRnodeExitCallback, VRno-deType, VRpan, VRstaticQuality, VRtilt, VRtriggerCallback, VRwarpMode
#richtext (obsolete in D7)	pageHeight, picture, scrollTop, text (authoring only) (other properties available for #field members are *not* supported)	See common properties.
#script	scriptText, scriptType	Scripts themselves are never sprites, but can be attached to sprites (see *the scriptNum, scriptList,* and *scriptInstan-ceList of sprite* properties for other sprite types).
#shape	filled, lineDirection,[3] lineSize, pattern, shapeType	blend, blendLevel, lineSize
#sound	channelCount, cuePointNames, cuePointTimes, loop, sampleRate, sampleSize	currentTime of sound,[7] mostRecent-CuePoint of sound,[7] volume of sound[7]

Table 4-10: Cast Member and Sprite Properties (continued)

Media Type	Cast Member Properties	Sprite Properties
#SWA[2]	bitsPerSample, bitRate, copyrightInfo, cuePointNames, cuePointTimes, duration,[5] numChannels, percentPlayed, percentStreamed, preLoadBuffer, preLoadTime, sampleRate, soundChannel, state, streamName, url, volume	currentTime,[7] mostRecentCuePoint,[7] volume[7]
#text[3]	alignment, alpha, antiAlias, antiAliasThreshold, autoTab, backColor, bgColor, border, bottomSpacing, boxType, charSpacing, color, dropShadow, editable, firstIndent, fixedLineSpace, font, fontSize, fontStyle, foreColor, lineCount, lineHeight, HTML, hyperlinks, leftIndent, lineSpace, kerning, kerningThreshold, margin, pageHeight, paragraph, picture, preRender, rightIndent, RTF, saveBitmap, scrollTop, selection, selectedText, tabCount, tabs, text, topSpacing, use HypertextStyles, wordwrap	editable, rect
#transition	changeArea, chunkSize, duration,[5] transitionType	N/A (never used as a sprite)
#vectorShape[3]	antiAlias, backgroundColor, closed, endColor, fillColor, fillCycles, fillDirection, fillMode, fillOffset, fillScale, gradientType, strokeColor, strokeWidth, vertexList See also #flash entry.	See also #flash entry.
#xtra	Xtra-dependent. See common properties.	Xtra-dependent. See common properties.

[1] See the *streamName of member* and *url of member* properties for SWA cast members.
[2] Requires an Xtra.
[3] New in D7.
[4] The *hilite of member* property does not apply to bitmaps and does not coincide with the *Hilight When Clicked* option in the Bitmap Cast Member Properties dialog box (there is no Lingo equivalent).
[5] The *duration of member* has different time units for digital video, SWA, and transition cast members.
[6] The *sound of member* is a Boolean property of digital video cast members. For any other type member, it simply returns (*sound memberNum*).
[7] Use the *currentTime of sound*, *mostRecentCuePoint of sound*, and *volume of sound* properties when referring to #sound cast members in the sound channels. Use the *currentTime of sprite*, *mostRecentCuePoint of sprite*, and *volume of sprite* properties when referring to #digitalVideo, #QuickTimeMedia, and #SWA cast members in the sprite channels.
[8] The *textAlign*, *textFont*, *textHeight*, *textSize*, and *textStyle of member* properties for field cast members are obsolete and have been replaced by the *alignment*, *font*, *lineHeight*, *fontSize*, and *fontStyle* properties.
[9] The Shockwave 6.0.1 plug-in does not support the *picture of member* property for fields.
[10] The author-time Flash Xtra's *showProps(member)* and *showProps(sprite)* methods list the Xtras' supported member and sprite properties.
[11] Use the *movieRate*, *movieTime*, and *volume of sprite* properties in D7, and the *mRate*, *mTime*, and *volumeLevel* of sprite properties in D6.5. The VR-related sprite properties apply only if the *isVRmovie of member* property is TRUE. In D7, the sprite properties beginning with "VR" are deprecated and replaced by properties of the same name with out the "VR" prefix, such as *fieldOfView, hotSpotEnterCallback*, and so on.

Cast member and sprite property idiosyncrasies

Two or more sprite properties may not always return consistent information, and some properties return the wrong information. There are a number of idiosyncrasies pertaining to specific properties:

CastType versus type and memberType

The obsolete *castType of member* property returns *#text* for field cast members, whereas the *type of member* property returns *#field* for the same cast members. In D7, the *#text* type was recycled. A *type of member* of *#text* now identifies a new asset type that replaces *#richText* in D7. The misdocumented *memberType of member* doesn't exist and shouldn't be used. The *type of sprite* property returns 0 for empty sprites, and 16 for all other sprites; use this to find the type of asset associated with a sprite:

```
put the type of the member of sprite whichSprite
```

Width and height of graphic sprites

When swapping cast members for a sprite, the *height* and *width of sprite* may not update properly. Use *the height of the rect of sprite x* and *the width of the rect of sprite x* instead.

Width and height of field and text sprites

Although the properties of each sprite instance are usually unique, the *width of sprite* and *height of sprite* properties can not be set independently for different sprites created from the same field or text cast member. All field or text sprites created from a single cast member use the same width and height.

Hilite of sprite for buttons

The *hilite of member* of a button can be set only on a cast member basis. There is no *hilite of sprite* property. You must create separate cast members to create independent buttons.

Video sprite properties

The *center*, *controller*, *crop*, or *directToStage* properties cannot be set for digital video *sprites*. These can only be set on a cast member basis. Duplicate the cast member to apply different member properties.

Rich text properties lacking Lingo access

Director 6 does not provide Lingo access to the rich text cast member properties that are accessible for fields, such as *alignment*, *font*, *fontSize*, and *fontStyle*. These attributes can be set for multiple selected rich text cast members using the Text Inspector, `Modify ➤ Font`, or `Modify ➤ Paragraph`. Additional attributes must be set individually using the cast member properties dialog box. See Chapter 12, *Text and Fields*. In D7, use new *#text* members for which runtime properties are settable.

Changing cast member and sprite properties

Table 4-11 lists the convenient places to alter different sprite and cast member properties. Changing sprite properties in the *prepareMovie* and *startMovie* handlers is not reliable.

Table 4-11: When and Where to Change Member and Sprite Properties

To Change Sprite Properties When:	Use These Types of Handlers:
Playback head enters a sprite span	*on beginSprite*
Playback head leaves a sprite span	*on endSprite*
Cursors rolls over sprite	*on mouseEnter*, *on mouseWithin*, and *on mouseLeave*
User clicks on sprite	*on mouseUp*, *on mouseDown*, *on rightMouseUp*, or *on rightMouseDown*
Before frame is drawn	*on prepareFrame* (or *on stepFrame* if sprite is included in *the actorList*)
After frame is drawn	*on exitFrame* (avoid *on enterFrame*)
No other events are being processed	*on idle*

When you puppet a sprite manually, its initial values are taken from the frame in which you issue the *puppetSprite* command. Avoid puppeting an empty sprite channel. Use an offscreen placeholder sprite, if necessary, and then set the *loc of sprite* to bring it on-Stage.

If you puppet an empty sprite channel, you must manually set the *width*, *height*, and *member of sprite* properties, and must often set the *loc* and *foreColor of sprite* properties, too.

In D7, the new *locZ of sprite* property (which defaults to the channel number but can be increased or decreased) changes the order in which sprites are layered. In prior versions, you cannot change a sprite's z-ordering directly. You could simulate it by swapping sprite properties with a sprite in a different channel. For example, to create a sprite that appears in front of all other sprites, you can set the properties of a placeholder sprite in the highest numbered channel.

Changing a Sprite's Properties Based on User Actions

It is common to modify a sprite's properties to make it respond to user actions. For example, you might change the cast member of a sprite when the user rolls over it or clicks the mouse. You should avoid hardcoding cast member names and sprite channel numbers, and instead create generalized handlers as described in Chapters 1 and 9 in *Lingo in a Nutshell*. You can use the Lingo properties *the currentSpriteNum*, *the clickOn*, *the spriteNum of me*, *the rollover*, *the member of sprite*, and *the memberNum of sprite* to create flexible handlers that will work when attached to any sprite in any channel.

Example 4-3 assumes that a highlighted and depressed version of the sprite's cast member are stored in the next two cast member positions. It highlights the button when the mouse rolls on the sprite and shows a depressed state when the mouse button is pressed. It handles the case where the users rolls on and off the sprite while holding the mouse down, and resets the sprite when the mouse rolls off or is released. Place this in a sprite script and attach it to a sprite.

Example 4-3: Multistate Button Behavior

```
property pOrigMember

on beginSprite
  set pOrigMember = the member of sprite (the currentSpriteNum)
end

on mouseEnter
  if the stillDown then
    set addCast = 2
  else
    set addCast = 1
  end if
  set the memberNum of sprite (the currentSpriteNum) = ¬
    the memberNum of pOrigMember + addCast
end

on mouseLeave
  set the member of sprite (the currentSpriteNum) = pOrigMember
end

on mouseDown
  set the memberNum of sprite (the currentSpriteNum) = ¬
    the memberNum of pOrigMember + 2
end

on mouseUp
  set the member of sprite (the currentSpriteNum) = pOrigMember
  go next
end
```

Cast Utilities

The following sections contain utilities that manage cast members.

Checking for duplicate cast member names

In Example 3-9, you saw how to cycle through every sprite channel of every frame of the Score. In this example, we cycle through each cast member of each castLib. This can be used as the basis for other utilities that perform some check on all the cast members.

Example 4-4 creates a Lingo list of all the cast member names. Using the *examineList()* utility from Example 3-2, we can check the cast member names for duplicates or potentially extraneous spaces:

```
examineList (buildCastmemberNamesList())
```

Example 4-4: Checking for Troublesome Cast Member Names

```
on buildCastmemberNamesList
  set nameList = []
  -- Create a list containing castmember names
  repeat with i = 1 to the number of castLibs
    repeat with j = 1 to the number of members of castLib i
      -- Find cast members with names
      set thisMember = member j of castLib i
      if the type of thisMember <> #empty then
        if the name of thisMember <> EMPTY then
           add (nameList, the name of thisMember)
        end if
      end if
    end repeat
  end repeat
  return nameList
end buildCastmemberNamesList
```

Finding linked cast members

The utility in Example 4-5 finds all cast members that have links to external files and reports if the specified path is not in the same folder as the current Director movie. Add clauses to the *case* statement as needed to handle the different properties for linked file paths.

Example 4-5: Checking for Linked Cast Members

```
on checkLinks
  -- Create a list containing castmember names
  repeat with i = 1 to the number of castLibs
    repeat with j = 1 to the number of members of castLib i
      set thisMember = member j of castLib i
      case (the type of thisMember) of
        #SWA: set linkPath = the streamName of thisMember
        #flash: set linkPath = the pathName of thisMember
        otherwise: set linkPath = the fileName of thisMember
      end case
      if linkPath <> EMPTY then
        if (linkPath starts the moviePath) then
          put "Linked" && thisMember & ":" && linkPath && "OK"
        else
          put "Linked" && thisMember & ":" && linkPath && ¬
               "not in same folder as DIR movie"
        end if
      end if
    end repeat
  end repeat
end checkLinks
```

Reader Exercise: Modify Example 4-5 to check whether the specified files exist and to verify that the file paths obey the Windows 3.1 file naming requirements. You might use an appropriate Xtra to copy the files to the local directory, and reset the *fileName, streamName,* or *pathName of member* property. You might even sort the

files into different subdirectories based on their media type. You can also modify the example to find and check linked castLibs using *the fileName of castLib* property.

External file sizes

The *size of member* property does not return meaningful data for most externally linked files. To obtain the actual size of an external file on disk, use Example 4-6, which returns the size in KB. It requires the FileIO Xtra and returns –43 as an error code if the file cannot be found and other negative numbers for other errors. (See Chapter 14 and Appendix E, *Error Messages and Codes*, in *Lingo in a Nutshell*.)

Example 4-6: Determining External File Sizes

```
on getSize extMember
  -- Returns the file size of an external asset, in KB
  -- This assumes that the FileIO Xtra is installed
  set fileObj = new (Xtra "FileIO")
  if objectP(fileObj) then
    -- Get the file's length
    openFile (fileObj, the fileName of member extMember, 1)
    set errCode = status (fileObj)
    -- A negative code indicates an error
    if errCode < 0 then
      return errCode
    else
      set fileSize = getLength (fileObj) / 1024.0
      set fileObj = 0
      return fileSize
    end if
  else
    -- Make up an error code if new() fails.
    return -1
  end if
end getSize
```

Importing linked cast members

The utility in Example 4-7 imports linked bitmap and sound cast members that are less than 1024 KB (1 MB) into the cast. It uses the *getSize()* utility from Example 4-6 to calculate the size of the external file. Note that it preserves the cast member name, which is ordinarily destroyed by *importFileInto*. It should be used during authoring, not at runtime, and requires the FileIO Xtra as well as the MIX Xtras to import any linked data types. I've used a *case* statement so that you can easily modify it to import other data types. See also Example 4-5.

Example 4-7: Importing Small Externally Linked Files

```
on importLinks
  -- This iterates through all members in all castLibs
  repeat with i = 1 to the number of castLibs
    repeat with j = 1 to the number of members of castLib i
```

Example 4-7: Importing Small Externally Linked Files (continued)

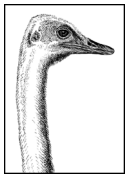
```
      set thisMember = member j of castLib i
      set linkPath = the fileName of thisMember
      if linkPath <> EMPTY then
        set size = getSize(thisMember)
        -- This only imports small #bitmaps and #sounds
        -- and does not import other data types
        case (the type of thisMember) of
          #bitmap, #sound:
            -- If no error and it is less than 1024 KB
            if size > 0 and size <= 1024
              put "Importing" && thisMember && linkPath
              -- importFileInto destroys name, so save it
              set oldName = the name of thisMember
              importFileInto (thisMember, linkPath)
              set the name of thisMember = oldName
            else if size = -43 then
              -- The file may have been too big to import. Print
              -- an error message if the file was not found.
              put "Couldn't find" && thisMember && linkPath
            end if
        end case
      end if
    end repeat
  end repeat
  put "Done Importing links"
end importLinks
```


Counting cast members

The *number of members of castLib* property reports the last used member, not the number of occupied cast member positions. Example 4-8 counts the actual number of used cast members. I've used a *case* statement so that you can easily modify it to count cast members of specific data types (you'll need additional variables besides *n*).

Example 4-8: Counting Cast Members in a castLib

```
on countMembers
  -- Count the number of occupied castmember slots
  repeat with i = 1 to the number of castLibs
    set n = 0
    repeat with j = 1 to the number of members of castLib i
      case (the type of member j of castLib i)
        #empty: nothing
        otherwise: set n = n + 1
      end case
    end repeat
    put "CastLib" && i && "has" && n && "cast members" && ¬
   "(last member:" && the number of members of castLib i & ")"
  end repeat
end countMembers
```



CHAPTER 9

Memory and Performance

You must balance the sometimes conflicting requirements of disk usage, download times, memory usage, and runtime performance. Compressed media shortens the download time of a Shockwave movie, but takes longer to decompress. Bitmaps require more memory, but draw more quickly than QuickDraw shapes. There is usually a trade-off between compression and quality for digital video, audio, and bitmaps. Although efficient use of memory often improves performance, the best balance depends on the minimum playback platform and the nature of your project.

See "Determining the Appropriate Minimum Hardware Playback Platform" at *http://www.macromedia.com/support/director/how/expert/playback/playback.html* for details on specifying the appropriate hardware (and convincing the marketing people to go along with you).

Disk Storage and Memory Management

When you save a movie file, Director saves only the changes made since the last save. The *movieFileFreeSize* indicates the size of old edited or deleted cast members remaining in the movie file. `File ➤ Save and Compact` rewrites the current data to a temporary file, excluding any old data. Director then deletes the original file and renames the temporary file to match the original file's name. `File ➤ Save As` also compacts the file. Both operations require enough disk space (at least temporarily) for the compacted file in addition to the original.

Director stores cast members in internal castLibs in the order in which they are used in the Score. Director stores cast members in external castLibs in the order in which they appear in the Cast window.

Each cast member is stored only once, although it may be used multiple times (or not at all) in the Score. For internal Casts, include cast members used solely as *puppetSprites* in a dummy Score frame near the frame(s) to which they relate. They will be stored to disk—along with the other nearby sprite's cast members—

although not loaded when the "live" frame loads. For external castLibs, use `Modify ➤ Sort ➤` *Usage in Score* before performing `File ➤ Save and Compact`.

A cast member's storage order is unrelated to whether it is *loaded* at runtime. By default, Director does not load cast members for frames it never reaches. To load *puppetSprites* along with a "live" frame, place them off-Stage in a sprite channel of the live frame. Director loads cast members for off-Stage sprites, but not for invisible or muted sprite channels (see *the visible of sprite* property and the mute buttons in the Score window).

Memory Management

Director does not usually load all assets into memory at once. The Director movie, external castLibs, and streaming digital video and audio files may be many times larger than the available memory. Director automatically loads cast members as needed and unloads old ones to make room for new cast members.

Test your presentation on your target platform before attempting to tweak the memory loading manually. You can control loading indirectly via *the preLoadMode of castLib* and *the purgePriority of member* properties or explicitly using the preloading, idle loading, and unloading commands described later in this chapter.

Memory Allocations

Windows and the Mac OS allocate memory very differently. This affects the memory available to Projectors and is reflected in several system properties and the Memory Inspector (check *the memorySize* to determine how much memory is allocated to Director or a Projector). Real RAM is much faster than *virtual memory* (VM, in which a *swap file* on the hard drive is used to simulate additional RAM). Director implements its own "virtual memory" scheme, swapping data from the disk to RAM as needed. It is counterproductive for Director to load a cast member from disk only to have it swapped back out to disk by the OS (although it will be accessed faster from a hard disk cache than from a CD-ROM). Tell your users to disable RAM Doubler and similar utilities that degrade performance and confuse Director's memory management.

Multimedia is memory- and processor-intensive. Real RAM is very cheap and strongly preferred. Virtual Memory should be *off* if possible on the Macintosh, but *on* under Windows, although Windows users should also have adequate available RAM.

Macintosh memory allocation

Macintosh applications, including Director and Projectors, request a fixed block of memory when they are launched and will fail to launch unless at least the minimum requested memory is available. Check the minimum and preferred memory allocations by highlighting an application's icon and choosing the Finder's `File ➤ Get Info` option. These can be edited only when the application is not running.

Increase the preferred memory allocation for Director to allow you to import more cast members during authoring, and to generally improve performance. The Macintosh-only `File ➤ Preferences ➤ General ➤` *Use System Temporary Memory* option allows Director to access additional system memory during authoring. You

can use this option unless it appears to cause conflicts on your particular development machine.

A D7 Macintosh Projector requests a minimum of 4,096 KB and a maximum of 6,144 KB, by default (see Table 9-1), but some Projectors will fail to launch if the full 6,144 KB is not available. Until it is fixed in D7.0.1, set the minimum allocation to 6,144 KB manually. Adjust these defaults according to your project needs and the limits of your target playback platform.

There is rarely reason to allocate more than about 12 MB to your Macintosh Projector. Allocating 50 MB, for example, will merely consume all available RAM, leaving no headroom for the Mac OS, which will cause problems.

Macintosh Projectors built using the `File ➤ Create Projector ➤ Options ➤` *Use System Temporary Memory* option can access additional system RAM beyond their fixed allocation. This setting is ignored when the Projector is played back on a Macintosh with Virtual Memory enabled. Avoid this option if supporting 68040 Macs. If the user has enough RAM, it is strongly recommended that VM be turned off using the *Memory* Control Panel on the Macintosh.

Macintosh Projector memory usage

The amount of memory your Projector will require depends on many factors, such as the monitor color depth and whether you are using digital video. By default, Fat Macintosh Projectors are allocated from 2 MB to 7.3 MB depending on the Director version, processor type, virtual memory setting, and the amount of memory available when the Projector starts. You can adjust the memory allocation manually using the Finder's *Get Info* command after the Projector is built. Mac OS X will adopt a similar approach to Windows, where applications do not use fixed partitions.

Table 9-1 shows how the default memory allocations for each type of Macintosh Projector vary by version and processor type. The minimum and preferred memory allocations adjust automatically when virtual memory is turned on or off. For Director 4.0.4, the minimum and preferred allocations were always 2 MB and 4 MB, respectively.

Table 9-1: Macintosh Projector Default Memory Allocation

Processor[1]	D7 Minimum/Preferred	D6.0.2 Minimum/Preferred	D5.0.1 Minimum/ Preferred
68K Mac	N/A	2,048/6,144 KB	2,048/4,096 KB
PowerMac with VM on	4,096/6,144 KB	2,048/6,1444 KB	2,048/4,096 KB
PowerMac with VM off	4,263/6,311 KB[2]	3,388/7,484 KB	2,986/5,034 KB

[1] A PowerMac running a Standard (68K) Macintosh Projector uses the default memory specifications for a 68K Mac regardless of its VM setting.
[2] The minimum and preferred memory allocations with VM turned off vary slightly depending on the Projector type and Xtras bundled within it.

Use *the memorySize* and *the freeBlock* to check how much RAM the Projector successfully allocated and how much remains available.

Windows memory allocation

Windows applications (including Director and Projectors) request memory as needed from a common system pool. Windows applications do not receive fixed allocations as do Macintosh applications. The Windows-only `File ➤ Preferences ➤ General ➤` *Limit Memory Size* option is used during authoring in D5 and D6 to simulate playback on a machine with less memory. It is not available in D7.

Unlike the Macintosh, under Windows, VM (a permanent swap file) should be enabled both during authoring and for Projectors. Some versions of Director can run without VM if more than 64 MB RAM is installed, but VM is required in most cases. D7 will fail to launch if insufficient disk swap space is available.

To configure virtual memory under Windows 95/98, double-click the *System* Control Panel (accessed via `Settings ➤ Control Panel` from the Start Menu). Choose the *Performance* tab, click the *Virtual Memory* button, and then choose *Let Windows manage my virtual memory settings*. See Macromedia TechNote #03516, "Windows 95 Multimedia Configuration."

Under Windows NT 3.5.1, virtual memory is configured using the *Virtual Memory* tab in the *System* Control Panel. Under Windows 3.1, virtual memory is configured using the *386 Memory* Control Panel and via the *CONFIG.SYS* file. Under Windows 3.1, set the swap file to *None* (if you have enough RAM) or a *Permanent* swap file of about 2 MB, but don't use a *Temporary* swap file.

The *DIRECTOR.INI* file includes two options affecting disk swap space used under Windows for Projectors and during authoring:

```
[Memory]
ExtraMemory = kilobytes
SwapFileMeg = megabytes
```

The *ExtraMemory* option determines the amount of swap space (in KB) a Projector should use at runtime and defaults to 400 KB. Increase this to allocate more swap space to the Projector. The *SwapFileMeg* option determines the amount of swap file space (in MB) to be used during authoring only. It defaults to zero (a special setting that requests disk space equal to half of available physical RAM). Increase *SwapFileMeg* to perhaps 20 MB to import more cast members before running out of memory. See Appendix D, *The DIRECTOR.INI and LINGO.INI Files,* in *Lingo in a Nutshell* for additional details.

Audio buffers

The Macintosh uses a fixed audio buffer of about 400 KB. This means that it buffers less than 3 seconds of CD-quality sound (176 K/sec), but about 18 seconds of 22 kHz, 8-bit, mono sound (22 K/sec). The length of Windows audio buffers can be set via the *DIRECTOR.INI* file. See Macromedia TechNote #03107, which includes some sound buffer size calculations.

Media Sizes

Media elements require a lot of *bandwidth* (capacity) to be stored, loaded, and displayed. The *throughput* (ability to transfer data) of the processor, hard drive,

CD-ROM, network connection, video card, sound card, memory, and Director itself determine whether playback will be instant or delayed, smooth or jerky. You must account for the *latency* (delay) intrinsic to some devices, especially Internet connections, and their limited bandwidth.

The following sections describe each type of asset and how to calculate its size. See *http://www.zeusprod.com/nutshell/glossary.html* for definitions of the words *loaded, preloaded*, *purged* or *unloaded, streamed* and *streaming*, *internal castLib*, *internal asset*, *external castLib*, *external asset*, *linked*, and *unlinked*.

Internal (unlinked/embedded) and external (linked) cast members are treated similarly whether they reside in internal or external castLibs. External assets are generally streamed and internal assets are generally loaded in their entirety into memory when they are needed.

There are four different aspects of an asset's size to consider:

The size of each cast member's header information

For each cast member, Director loads a small header that describes its contents when the movie is first loaded. This header is completely separate from the media for the cast member, which may not be loaded until later. An excessive number of cast members (more than several thousand) can require significant RAM. Although media elements also require a lot of RAM, a cast member's media can be purged, but its header cannot. Likewise, the Score notation, shape cast members, and script cast members are all loaded when a movie or castLib is opened and are not purged until the movie ends. In D7, font cast members are also always loaded.

The size of a loaded asset's media in memory

The size shown in the Cast Member Info window (and by *the size of member* property) loosely indicates the amount of memory a cast member occupies *if* it is loaded (see *the loaded of member*). For cast members that point to other assets, such as film loops, *#digitalVideo*, and linked sounds, it represents the size of the cast member overhead or the *#digitalVideo*'s header data. In such cases, the true size of the asset is usually much bigger than shown. The *size of member* reflects the disk file size for *#QuickTimeMedia* members.

Select multiple cast members and use `Modify ➤ Cast Member ➤ Properties` to view their cumulative size.

The size of an asset on disk

Whether in an internal or external castLib, or an external file, the time to load or download an asset is determined by its compressed size on disk which is smaller than its size once loaded into memory. There is no easy way to determine the size of internal cast members compressed on disk (see Example 9-1).

The data rate of streaming media

It is possible to play very large external video and sound files that exceed the available memory because they are streamed from disk in "chunks" that are discarded once played. When streaming data, the main concern is not the

entire file's size, but its *data rate* (the amount of data per second that must be loaded). For example, a video compressed to 1 MB/sec requires more throughput than one compressed to 400 KB/sec. Likewise, a CD-quality audio file requires 176 K/sec of data, versus a 22 kHz, 8-bit, mono sound requiring only 22 K/sec. The data rate of *uncompressed* streaming media (standard audio files) depends on the characteristics of the original content (sample rate, number of channels, and so on). The data rate of *compressed* streaming media (digital video and Shockwave audio) is primarily determined by the compression settings and the desired fidelity.

Streaming data

The `Modify ➤ Movie ➤ Playback` option lets you specify how Director should handle streaming Internet media from within the Shockwave plug-ins within a browser. This also affects the playback of cast members at a remote URL linked into a Projector (streaming options vary slightly in D6 and D7).

Bitmaps and PICTs

Embedded (unlinked) bitmaps are converted to Director's internal bitmap format (unless imported as a PICT). Director's internal format is optimized for a balance between disk size and access speed. It uses *RLE* (Run-Length Encoding) compression. Large areas of the same color compress extremely well, and the number of unique colors in a graphic determines its size on disk. (A 16-bit graphic using 256 colors will compress to the same size as an 8-bit graphic using 256 colors.)

Once a bitmap is loaded, its uncompressed size in RAM can be much larger than the disk storage size. This is calculated (in bytes) as:

(the width of member)× (the height of member)× (the depth of member)/8

Thus, an 8-bit graphic uses one-quarter the RAM of a comparable 32-bit graphic. Bitmaps at a different depth than the monitor must be converted on the fly, which slows performance.

The RAM used by a bitmap depends on its bounding rectangle, so an L-shaped graphic that is 300 pixels on each side takes up the same RAM as a solid 300 × 300 graphic. Cut L-shaped graphics into two cast members to reduce the memory required by upwards of 75%. Likewise, a four-sided framing graphic with a large blank center would occupy much more RAM than four individual sides of the frame.

Linked bitmaps in formats such as JPEG and GIF tend to be smaller on disk but much slower to load, and occupy the same memory once loaded as another bitmap. Linked bitmaps are often so slow as to be unusable. I import bitmaps as unlinked even if I expect the artwork to change. PICT cast members retain their original PICT format and typically require less disk space and less memory, but are slower to load than standard bitmaps. D7 also supports internal JPEG- and GIF-compressed cast members. Import them using the *Include Original Data for Editing* option.

When creating animations, consider the number and size of bitmaps you will need over time and how fast they can be loaded from disk or the Internet.

Shapes, Vector Shapes, and Flash

QuickDraw shape cast members are incredibly efficient and occupy only 64 bytes, as indicated by *the size of member* property. Shapes using a fill pattern or custom tile are much more compact than an equivalent bitmap, although they take slightly longer to draw. A single shape can be stretched and colorized to create multiple sprites. Shapes are always loaded in memory, but this overhead is usually minimal.

Flash and D7's new vector shape cast members are vector-based graphics that require extremely low storage and RAM, but more processor power. Flash and vector shape members are supported on Win32 and Mac PPC systems only.

For information about the Flash Asset Xtra see the HTML help files included with D6.5, the D7 Help, Table 4-10, and *http://www.zeusprod.com/nutshell/appendices/flash.html*.

Buttons

There are two entirely distinct button types:

Standard buttons
: Built-in Director push button, check box, and radio button cast members created with the Tool Palette occupy only about 250 bytes each.

Custom buttons
: Custom Buttons are inserted via `Insert ➤ Media Element ➤ Custom Button` in D6. A Custom Button can contain up to eight states, each using a different graphic. The *size of member* property for Custom Button cast members depends on the size of the underlying bitmaps incorporated into it. Unlike film loops, those assets can be deleted once they are incorporated into the Custom Button. Leave unused button states empty to conserve memory. The Custom Button Xtra is obsolete in D7.

Fields

Field cast members are limited to 32,000 (not 32,768) characters in D6, and occupy between about 250 bytes and 35 KB (the character and size limit is eliminated in D7). This includes the cast member header, plus one byte per character, plus overhead for formatting (about 25 characters per style run).

To calculate the length of a field's contents, use:

```
put length (field whichFieldMember)
put the length of the text of member whichFieldMember
```

Field string manipulation can become egregiously slow if a string contains more than a few thousand characters. Copy the contents of a field to a string variable to perform string manipulations, then copy the result back to the field. String variables can contain strings up to about 2 MB, but they too can become slow at those sizes.

Formatted and colorized fields can be slow and may occupy much more memory than unformatted text. Keep the formatting simple.

Rich Text and Text

Rich text cast members in D5 and D6 are stored as *bitmaps*, and their *size of member* property depends mainly on the number of characters and point size. A typical rich text cast member may be from 2 KB to more than 200 KB (25 times larger than a comparable field cast member). Rich text cast members are not compressed when converted to Shockwave format. Prior to D7, convert rich text cast members to bitmaps for better compression, use field cast members (which are always smaller), or use the Flash Asset Xtra, which provides high-quality animated text at extremely low bandwidths. In D7, use the new text and font cast members (as described in Chapter 12), which are space-efficient.

Film Loops and Movie Cast Members

The size of member property for film loops is usually about 1 KB and does not include the cast members that make up the actual film loop. To determine their size, copy the film loop into the Score, and use the *ramNeeded()* function to determine the memory required for the range of frames comprising the film loop. See "Film Loops" in Chapter 3.

The *size of member* property for linked movie cast members is zero and does not reflect the size of the external movie file.

Palettes and Transitions

By using 8-bit custom palettes, you can reduce the RAM required for bitmaps substantially (see Chapter 13). Cast members using built-in transitions require 0 extra bytes. Third-party transition Xtras usually occupy very little memory, although some that use precalculated data may be larger.

Scripts

Script cast members are limited to 32,000 (not 32,768) characters in D6 (this limit is removed in D7), but their *size of member* property may be twice that (about 60 KB), because it includes the size of the compiled script. The original *scriptText* is stripped out of protected movies, which reduces the scripts' size by about half.

During authoring, you can calculate the length of a script's contents using:

```
put length (the scriptText of member whichScriptMember)
```

All the script cast members in a movie and its external castLibs are always loaded and never swapped out until the movie closes. Hundreds of scripts may occupy substantial memory.

Generalize your handlers or use Behaviors to reduce the number of scripts in a project. Refer to Chapter 1, *How Lingo Thinks,* and Chapter 12, *Behaviors and Parent Scripts*, in *Lingo in a Nutshell* for details.

Digital Videos

Digital videos are always externally linked and streamed from disk as they play, enabling a large digital video file to be played without requiring excessive memory. *The size of member* reflects only the size of a *#digitalVideo* member's header, but

reflects the true external file size for *#QuickTimeMedia* members. Digital videos should be compressed using MediaCleaner (formerly MovieCleaner), Adobe Premiere, or a similar utility.

A video's average and peak data rates affect performance much more than the total overall size on disk. A video's average data rate can be calculated as:

$$\frac{\textit{(size of the external digital video file)} \times \textit{float(the duration of member)}}{\textit{(the digitalVideoTimeScale)}}$$

A video's peak data rate is also of concern, although it is often not significantly higher than the average data rate. Use an external tool such as Adobe Premiere to check a movie's peak data rate. When budgeting for bandwidth, don't forget the size of the audio track(s) within the digital video file.

Sounds

A sound's memory requirements depend on its fidelity, whether it is compressed, and whether it is linked (external) or unlinked (internal). The size of a sound depends partially on the number of channels within the sound (1 for mono, 2 for stereo). See Chapter 15 for additional details and caveats about each type of sound.

Linked external sounds

Externally linked sounds are streamed from disk as they play, enabling a large sound file to play without waiting for it to load and without requiring excessive memory. Only enough memory to buffer the sound is required (usually less than 400 KB). *The size of member* as reported for linked sounds does not accurately reflect the size of the external sound file.

Unlinked internal sounds

Internal embedded sounds are always loaded into memory before being played, and are best limited to small (less than 500 KB) sounds. Large sounds should be externally linked, instead. Use looping sounds to reduce memory requirements. *The size of member* accurately reflects an internal sound's size and can be calculated (in bytes) as:

$$\textit{(samples per second)} \times \textit{(bits per sample/8)} \times \textit{(length in seconds)} \times \textit{(number of channels)}$$

For example, an 11 kHz, 8-bit, mono sound requires 11 K/sec, and a 44 kHz, 16-bit, stereo sound requires 176 K/sec.

Compressed sounds

Director supports IMA-compressed AIFF audio (4:1 compression). IMA-compressed sound cast members are not further compressed by Shockwave, even when Shockwave audio compression is activated. If Shockwave audio compression is disabled, LZW compression is used for other internal sounds in DCR and CCT files (about 30% savings).

Shockwave audio (SWA)

Naturally, Shockwave audio (SWA) compression is used for external SWA files, but it can be used for internal sound cast members with both Projectors and Shockwave (see the `Xtras ➤ Shockwave for Audio Settings` option). With SWA, you choose an output bit rate (target bandwidth), not a compression ratio. An SWA sound's download size can be calculated (in KB) as:

$$\frac{\textit{(the bitRate of member)}}{\textit{(8.192)} \times \textit{(the duration of member)}}$$

The MPEG-3 compression algorithm used by SWA yields higher quality sound without additional bandwidth when using a higher fidelity source. Always use either 22 kHz or 44 kHz 16-bit source audio for SWA compression.

Note that SWA sounds requires measurable processing power and may hinder performance on lower-level machines.

Xtras

The amount of memory required for Sprite Asset Xtras is highly dependent on the asset type. The Flash Asset Xtra provides excellent vector-based graphics at extremely low bandwidths and allows fine control over its memory use. Other Xtras, such as the QT3 Xtra, may use substantially more memory and/or use memory-intensive data types.

Determining Asset Sizes via Lingo

The *size of member* property reliably returns the RAM required for most internal cast members, including bitmaps, internal sounds, shapes, buttons, scripts, fields, and text cast members. It does not accurately reflect the RAM used by external sounds (SWA included), digital video, film loops, and movie cast members. Table 9-2 shows cast member properties useful in determining an asset's size.

You can check the size of an external file in the Finder or File Explorer during authoring. Refer to Example 4-6, which calculates an external file's size.

Table 9-2: Size-Related Member Properties

Media Type	**Member Size**
#bitmap	*the depth*, *height*, *width*, and *size of member*
#button	*length(the text of member)* and *the size of member*
#digitalVideo	*the duration*, *frameRate*, and *preLoad of member*, and *the preLoadRAM*; see Example 4-6
#field	*length(the text of member)* and *the size of member*
#filmLoop	*the media of member* (must be unwrapped)
#movie	see *getSize()* utility in Example 4-6

Table 9-2: Size-Related Member Properties (continued)

Media Type	Member Size
#picture (imported as PICT)	see *getSize( )* utility in Example 4-6
#QuickTimeMedia	*the size of member*; see the *getSize( )* utility in Example 4-6
#richText	*the size of member*
#script	*length(the scriptText of member)* and *the size of member*
#sound (linked)	*the channelCount*, *sampleRate*, and *sampleSize of member*
#sound (internal)	*the size of member*
#swa	*the bitsPerSample*, *sampleRate*, *numChannels*, and *duration of member*
#text	*length(the text of member)* and *the size of member*

Data Throughput

When calculating the acceptable size of an asset, keep in mind the speed of the device (such as a CD) from which it will be loaded. The practical data rate is somewhat lower than the theoretical data rate for a CD-ROM. Table 9-3 shows the approximate time to load 5 MB of data from various CD-ROM drives. Remember that data is often compressed on disk, so that a 300 KB bitmap may require less than half that on disk. Thus, even a quad-speed CD-ROM may be able to load two full-screen (640×480×256-color) images per second. In practice, there may be some latency when first accessing data, due to Director having to find it on the CD-ROM.

CD-ROM performance can also depend on the driver and cache settings. The third-party CD-ROM Toolkit (by FWB) can alter Macintosh CD-ROM drive settings. Under Windows 95, see the *System* control panel (*Performance* tab ➤ *File System* button ➤ *CD-ROM* tab).

Table 9-3: CD-ROM Speeds

CD-ROM Drive	Theoretical Data Rate	Practical Data Rate	Load Time (5 MB)
Single-speed (1X)	150 KB/sec	100 KB/sec	50 sec
Double-speed (2X)	300 KB/sec	200 KB/sec	25 sec
Quad-speed (4X)	600 KB/sec	450 KB/sec	11 sec
Eight-speed (8X)	1200 KB/sec	900 KB/sec	4.3 sec
High-speed (>16X)	> 2400 KB/sec	> 2000 KB/sec	< 2.5 sec

Disk Capacity Budget

A typical CD-ROM holds about 650 MB of data. Table 9-4 shows how much data of a given type will fit on a single CD, but in practice, you will have a mix of various data types, plus some overhead for the installer, Projectors, Xtras, and so on. When using DVD-ROMs, which hold 4 GB or more, you can scale these figures accordingly. For many details on a variety of disc formats (and capacities), see the technical notes from Cinram at:

http://www.cinram.com/Techlibrary/technical_library.html
http://www.cinram.com/PDF/capacity.pdf

Table 9-4: CD-ROM Capacities

Asset Type	**Storage Requirement**	**Fits on CD**
Digital video (Cinepak—quarter screen)	400 KB/sec	27 minutes
Digital video (Cinepak—full screen)	1600 KB/sec	6.9 minutes
Digital video (Sorenson)	80 K/sec	135 minutes
MPEG-1 full-motion video	150 K/sec	74 minutes
MPEG-2 full-motion video	575 KB/sec	20 minutes
Audio (16-bit, 44.1 kHz, Stereo)	176 K/sec	64.5 minutes
Audio (16-bit, 22.050 kHz, Mono)	44 K/sec	4.3 hours
SWA (MP3) CD-quality	20 K/sec	9 hours
Bitmaps (640 × 480 × 256-color Director internal format)	200 KB/image[1]	3,300 images
Bitmaps (640 × 480 × millions of colors; JPEG compressed)	75 KB/image[1]	8,875 images

[1] A subjective approximation based on typical images with typical compression.

Director Memory Budget

Projectors require less memory than Director because they don't support many authoring-time features. Macromedia's Tech Note #03107, "Projector Memory Requirements," is woefully outdated and has several errors. The default *preferred* Macintosh Projector memory allocation may be adequate (see Table 9-1), but the default *minimum* memory allocation rarely is. Windows Projectors like to allocate at least 10 MB if it is available. Table 9-5 shows a possible memory budget for the Projector, exclusive of media elements. D7 recommends a minimum of 12 MB of RAM available for both Macintosh and Windows Projectors. I recommend a minimum 32 MB of real RAM installed, plus the virtual memory settings described earlier in this chapter.

Table 9-5: Program Memory Budget

Item	Mac	Win
Operating system[1]	5 to 10 MB	2 MB for Windows 3.1 8–24 MB for Windows 95/98/NT
Projector code	1 to 3 MB	1 to 3 MB
Projector misc. memory	1 to 3 MB	1 to 3 MB
Offscreen buffer[2]	300 KB or higher	300 KB or higher
Digital video drivers[3]	500 KB to 1.5 MB	500 KB to 1 MB
Xtras[4]	100 KB/Xtra	100 KB/Xtra

[1] The size of the operating system depends heavily on the extensions loaded.
[2] A 640 × 480 × 256-color (8-bit) Stage requires a 300 KB offscreen buffer.
[3] The size of QuickTime on the Macintosh varies with the version and the QuickTime plug-in components installed and used. For example, the first addition of a QuickTime cast member increases the System memory usage by about 650 KB. Under Windows, it is possible, though unusual, to have a project that uses both Video for Windows and QuickTime for Windows.
[4] This is a very rough estimate, but each Xtra regardless of its type (Scripting, Sprite Asset, Transition, MIX, etc.) consumes a small amount of RAM. Ship only the Xtras you need with your Projector.

Browsers often require 15 MB of memory or more. See Table 11-2.

Table 9-6 outlines the memory requirements that depend directly on media usage. These are arbitrary numbers based on a typical project. You can estimate the RAM needed for your product by performing calculations as shown under "Media Sizes" earlier in this chapter. Internal bitmaps and sounds are usually the biggest consumers of memory. The QuickTime 3 Asset Xtra requires substantial additional RAM. Macintosh Projectors using QT3 may need 15 MB. Your actual requirements may vary widely depending on the nature of your project. Remember that Director will load and unload cast members as needed, so you can survive with less memory at the expense of performance. In extremely low memory, Director may drop out sound or graphics.

Table 9-6: Media Memory Budget

Item	Mac	Win
Score notation and cast member headers	100 KB to 1 MB	100 KB to 1 MB
Script cast members	100 KB to 1 MB	100 KB to 1 MB
Bitmaps and other cast members	2 MB to 3 MB	2 MB to 3 MB
Internal sound cast members	500 KB to 1 MB	500 KB to 1 MB
Digital video[1]	500 KB to 1 MB	500 KB to 1 MB

Table 9-6: Media Memory Budget (continued)

Item	Mac	Win
Streaming buffer for external sounds[2]	400 KB	2.5 × the size of one second of audio (27 KB to 440 KB)
MIAW[3]	500 KB to 1 MB	500 KB to 1 MB

[1] Per typical 400 KB/sec video played concurrently. If *the preLoad of member* is enabled, see *the preLoadRAM*.
[2] Per concurrent sound streamed.
[3] Arbitrary estimate per MIAW. Exact RAM depends on size and complexity of MIAW.

Offscreen Buffer

Director composites sprites in an offscreen buffer whose size depends on the dimensions of the Stage. The size of the offscreen buffer can be calculated (in KB) as:

$$\frac{\textit{(the width of the rect of the stage} \times \textit{the height of the rect of the stage)}}{\textit{1024} \times \textit{(the colorDepth/8.0)}}$$

For example, a 640×480×8-bit offscreen buffer requires a 300 KB. In D6, if *the fullColorPermit* is `FALSE`, the size of the offscreen buffer is treated as if *the colorDepth* is 8-bit. Millions of colors is considered 32-bit, though *the colorDepth* reports it as 24-bit under Windows. Larger Stage dimensions and higher color depths usually imply that bitmaps will require more RAM as well.

MIAWs also increase the size of the offscreen buffer. In D6, it appears that MIAWs share an offscreen buffer with the Stage under Windows but have their own offscreen buffer on the Macintosh. In D7, the Stage and MIAWs have separate offscreen buffers on both platforms.

Cast and Score data

Although the Score's notation is fairly compact, the entire Score is loaded into memory when a movie is loaded. The size of the Score data also depends on the number of sprite channels used and the frequency of changes in the Score. Join sprites and eliminate unnecessary keyframes to reduce the Score notation's size markedly (you can save 1 MB over a large, inefficient Score).

There is also overhead associated with each cast member and their thumbnails (although the latter are stripped out when protecting a movie). Split movies containing thousands of frame changes or cast members into multiple movies to reduce the RAM used for the Score and Cast during the life of a given movie.

Data Structure Memory Requirements

Lingo variables and the data they point to require varying amounts of memory, as shown in Table 9-7. Simple types (integers, VOID values, symbols, and floats) always occupy 8 or 16 bytes. Complex types (strings, lists, child objects, and Xtra instances) occupy 8 bytes *plus* additional memory that varies with the size of the structure (such as the number of characters in a string, or the number of elements in a list). You can free the memory used by a complex data type by setting it to simple value, such as `VOID`, but even a `VOID` item will occupy 8 bytes.

Table 9-7: Lingo Data Structure Memory Requirements

Item	Minimum Size	Max Size
Integer	8 bytes	8 bytes
VOID	8 bytes	8 bytes
Symbol[1]	8 bytes	8 bytes
Float	16 bytes	16 bytes
String	8 bytes + 1 byte per character	About 2 MB
List	8 bytes + size of elements	About 2 MB
Xtra instance	8 bytes + 180 bytes	No specific limit
Child object (script instance)	8 bytes + 180 bytes	No specific limit

[1] Symbols always persist for the life of Director or the Projector. See *Lingo Symbol Table Archaeology* at *http://www.zeusprod.com/nutshell/chapters/symtable.html*.

Disposing of Objects (Freeing Memory)

All your variables combined may occupy less memory than a single bitmap. However, complex structures such as objects, strings, and lists can consume considerable memory and should be disposed of when no longer needed. Different types of objects are disposed of (freed) in different ways:

Variables

Local variables (those used within a single handler) are allocated when the handler is called and freed automatically when the handler terminates.

Property variables are allocated when the object, such as a Behavior script, parent script, or Xtra instance, is instantiated. They are freed when no variables refer to the object.

Global variables persist indefinitely, but by assigning a global variable to `VOID`, it occupies minimal memory. Avoid *clearGlobals*, which indiscriminately clears all globals as well as *the actorList* in D6 and D7. Use D7's new *the globals* property as described at *http://www.zeusprod.com/nutshell/d7/globals.html* to see a list of all globals currently allocated.

XObjects

Use *mDispose* to dispose of XObject instances and *closeXlib* to close XObject libraries. These are not for use with Xtras and are not supported in D7.

Xtra instances, child objects, lists, and strings

Set the variable pointing to the object to zero or `VOID`, such as:

```
set myInstance = 0
```

MIAWs

Use *forget window* to eliminate a MIAW from memory. *Close window* merely hides the window and does not release its memory. Clear any global variables, properties, and objects in use by the MIAW before disposing of it.

Movies accessed via play movie

Use *play done* to return from a movie that was accessed using *play movie*. The second movie's memory is not released until you return to the first movie using *play done*. Use *go movie* instead to reduce memory usage, as it immediately releases the old movie from memory.

Purgeable Items

Director loads and unloads many entities without your knowledge or instruction. The following items are purgeable if Director needs the memory:

- Cast members with the highest *purgePriority* that are not needed in the current frame are generally purgeable (exceptions follow).
- Objects no longer referred to by any variable can be purged. Thus, a list can be disposed of if no variables refer to it any more.
- Streaming video and sounds (including SWA) are immediately purged after each segment is played. Internal assets are not.
- Forgotten MIAWs will be purged (and disappear from *the windowList*).

The following items are never purged:

- Objects (such as Xtra instances, child objects, lists, and strings) that are still referenced by some variable
- The 8 bytes minimum required for each global, even if set to `VOID`
- Symbols (see Chapter 19, *The Lingo Symbol Table*, in *Lingo in a Nutshell*)
- MIAWs remaining on *the windowList*, whether visible (open) or not

The following items are not purged until leaving the current movie:

- The Score notation for the movie and cast member header information for each open castLib
- Script, Shape, and Transition, and new D7 Font cast members
- Active puppetSprites
- Cast members used in the current frame (unless RAM is unavailable even after unloading all purgeable assets)
- Cast members with *the purgePriority of member = 0* (*Never*)
- Cast members that have been imported, created, or modified since the Director movie was last saved (see *the modified of member*)

Cast Member Loading and Unloading

Cast members must be loaded from the disk or the Internet before they can be used. There is always a performance "hit" (delay) when cast members are loaded. You can either manually preload the cast members (and tolerate the delay) before the animation starts or let Director load the cast members as they are needed (and tolerate multiple small pauses as the animation plays).

Optimization of loading will not help if you are demanding too much throughput—reduce your media requirements instead!

Implicit Loading and Unloading

You can control cast member loading in many ways, but regardless, Director will attempt to load cast members when it needs them to draw the current frame. Use `Modify ➤ Cast Properties` or `Modify ➤ Movie ➤ Casts ➤` *Properties* to control cast member loading on a castLib basis. (These equate to *the preLoadMode of castLib* property.) The default setting (*When Needed*) loads cast members on demand, whereas the *Before Frame One* and *After Frame One* modes attempt to preload as many cast members as possible (use these mainly for linear presentations on dedicated hardware).

Automatic cast member unloading and the purgePriority

Director unloads the least recently used cast members as required to make room for new ones. Other unused cast members may remain loaded in case they are needed at a later time. The *Unload* option in the Cast Member Info dialog box (corresponding to *the purgePriority of member*) controls cast member unloading. *Normal* items with the highest *purgePriority* (3) are purged *first*, followed by items flagged as *Next* (*purgePriority = 2*), and finally *Last* (*purgePriority = 1*).

Don't use the *Never Unload* option (*purgePriority* = 0). It prevents Director from unloading assets even in desperate situations and can cause a crash.

Despite what Macromedia's older manuals and most third-party books erroneously imply, there is no "purge first" setting that purges cast members before *Normal* items. You must use *unLoadMember* explicitly to purge a cast member before other items.

The *Unload* setting is largely irrelevant for streamed assets, such as digital video and externally linked sounds, which are always discarded from memory as they are played. It is ignored for script, shape, transition, and font cast members, which are never unloaded.

Explicit Unloading

You can explicitly unload cast members using the commands in Table 9-8. The *unload* commands attempt to unload the cast members used in one or more frames of the Score, and the *unLoadMember* commands attempt to unload the specified cast members, but they may not be able to unload some of them for the reasons listed earlier. *UnloadMember* is buggy in D7.0, but fixed in D7.0.1.

Table 9-8: Unload Commands

Command	Usage
unload	Unloads only those cast members used in the Score.
unload *fromFrame* {, *toFrame*}	Unloads cast members in a range of frames, or in a single frame, if *toFrame* is omitted.
unload member *fromMember*, *toMember*	Unloads a range of cast members, as would the *unLoadMember* command.
unloadCast	Obsolete. See *unLoadMember*.
unLoadMember	Unloads all cast members in D6. In D7.0, but not D7.0.1, you must manually specify a range.
unLoadMember member *fromMember* {of castLib *fromCast*}, {member *toMember* of castLib *toCast*}	Unloads a range of cast members or a single cast member if *toMember* is omitted[1] (*fromCast* and *toCast* can be different castLibs).
unLoadMovie *whichMovie*	Unloads the specified movie (which can be a URL reference). *The result* returns 0 if successful or –1 if movie was not loaded.

[1] *UnLoadMember* requires that *fromMember* and *toMember* be valid member names or numbers. Use *the number of members of castLib* property to find the last valid member.

Reader Exercise: Write your own utility to unload a list of cast members or create an "exclusive unload" utility to unload all cast members *except* those specified.

Analyzing Memory Usage and Cast Member Loading

Director can analyze memory usage and cast member loading to help you track and debug memory problems.

The Memory Inspector

`Window ➤ Inspectors ➤ Memory` opens the Memory Inspector windoid (shown in Figures 9-1 and 9-2), which shows the memory allocated to various uses and includes a *Purge* button that frees as much RAM as possible. Refer also to the *Memory Inspector* entry in the online Help. The appearance of the Memory Inspector varies across platforms and depends on the *Use System Temporary Memory* (Mac) and *Limit Memory Size* (Windows) options under `File ➤ Preferences ➤ General`. The values reported in the Memory Inspector are not always reliable, and the area of the bars in the graph are not necessarily to scale.

To determine the installed RAM, virtual memory, and available RAM accurately, use a third-party Xtra, such as Buddy API's *baMemoryInfo()* method or OSutil's *OSGestalt()* method.

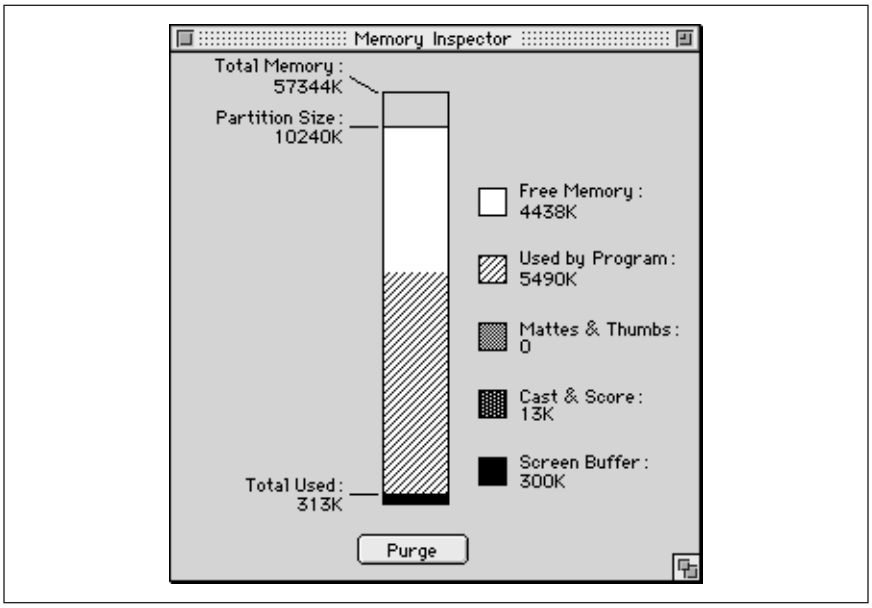


Figure 9-1: Memory Inspector (Macintosh)

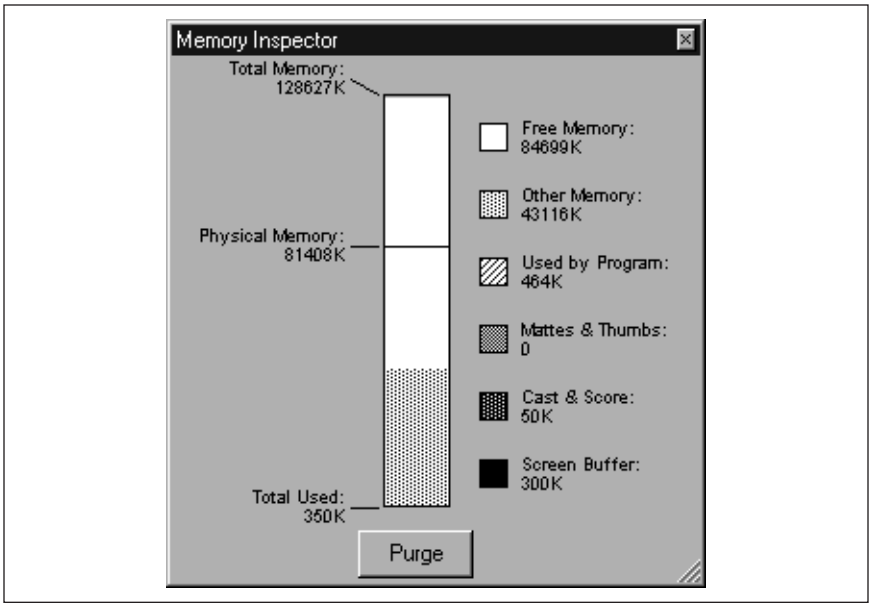


Figure 9-2: Memory Inspector (Windows)

The Memory Inspector displays the following values:

Total Memory
Installed physical RAM, plus virtual memory (if any).

Partition Size (Macintosh only)
Memory allocation set in the Finder's `File ➤ Get Info` window. Memory usage can exceed this if *Use System Temporary Memory* is checked under `General` preferences (or under `File ➤ Create Projector` options).

Physical Memory (Windows only)
Installed physical RAM only. Director can use virtual memory beyond the physical RAM.

Memory Limit (Windows only; not shown in Figure 9-2)
Reflects *Limit Memory Size* setting, if any, under `General` preferences.

Total Used
RAM currently in use for the offscreen buffer, Cast and Score notation, and mattes and thumbnails. This is *not* the total memory used by Director.

Free Memory (see the freeBytes in Table 9-9)
Unused memory available to Director. On the Macintosh, without *Use System Temporary Memory* enabled, it shows the unused portion of Director's fixed memory partition. On Windows, and on the Macintosh only when *Use System Temporary Memory* is enabled, it shows available system memory (including virtual memory).

Other Memory (shown on Windows, and on Mac if Use System Temporary Memory preference is checked)
RAM used by OS and other programs.

Used by Program
RAM currently used by Director. The value shown under Windows is completely wrong in D6.

Mattes & Thumbs
RAM used to create mattes and display thumbnails.

Cast and Score
RAM used to hold cast member and Score notation and edited cast members until they are saved to disk.

Screen Buffer
RAM used for offscreen compositing buffer. Size depends on *the colorDepth* and the Stage's dimensions.

Purge Button
Use this button to unload any items that are purgeable.

Determining Whether the Necessary Memory Is Available

Table 9-9 lists the commands that analyze RAM and disk space.

Table 9-9: Memory and Disk Space Analysis Commands

Command	**Usage**
cacheSize *newSize* put cacheSize( )	Gets or sets the cache size for downloadable media in a Projector or during authoring. Does not apply to Shockwave.
the fileName of member *whichMember*	Use this to locate the external file, whose size can then be determined using the FileIO Xtra. See Example 4-6.
frameReady ( )	Returns TRUE if the cast members required for the entire movie have been downloaded, or are local.
frameReady (*startFrame* {, *endFrame*})	Returns TRUE if the cast members required for the specified frame or range of frames have been downloaded, or are local.
the freeBlock	Returns size (in bytes) of the largest contiguous block of RAM.
the freeBytes	Returns total size (in bytes) of RAM available to Director, including Temporary System memory and virtual memory, if applicable.
getStreamStatus (netID or URL)	Get status of specified netID or URL (new in D7).
the loaded of member *whichMember*	Returns TRUE if cast member is currently loaded.
the mediaReady of member	Returns TRUE if cast member has been downloaded, or is local.
the memorySize	Returns size (in bytes) of the RAM allocated to Director or the Projector. Additional memory may be available (see *the freeBytes*).
the modified of member *whichMember*	Returns TRUE if cast member was created or modified since movie was last saved.
the movieFileFreeSize	Returns disk space (in bytes) that can be recovered using File ➤ Save and Compact.
the movieFileSize	Returns the size on disk in bytes of the current movie. Does not include external castLibs.
netDone(*netID*)	Determines whether a network operation, including *preloadNetThing*, has completed.
the purgePriority of member *whichMember*	Affects Director's automatic unloading of cast members. Those members with a higher *purgePriority* are unloaded earliest.
ramNeeded (*fromFrame*, *toFrame*)	Returns amount of RAM needed for cast members in the given range of frames. Use the same starting and ending frame number to check ramNeeded for a single frame.

Table 9-9: Memory and Disk Space Analysis Commands (continued)

Command	Usage
the size of member *whichMember*	Returns size of a cast member in RAM once it is decompressed from disk. Does not accurately reflect size of externally linked cast members.
the state of member *swaMember*	Returns "ready" if an SWA cast member has downloaded successfully.
tellStreamStatus (*flag*) put tellStreamStatus()	If *flag* is TRUE, Director will call the *on streamStatus* handler periodically, but if *flag* is FALSE (the default), it will not. If *flag* is omitted, *tellStreamStatus()* returns the current setting. See *getStreamStatus()*.
set the traceLoad = 0|1|2	Determines whether cast member loading is shown in the Message window. Highly improved in D7.
set the traceLogFile = *fileName* | EMPTY	Setting *the traceLogFile* directs the output from the Message window to an external text file. Setting it to EMPTY closes the external file.

The FreeBytes, the FreeBlock, and RamNeeded()

If *the freeBlock* is less than *the freeBytes*, RAM has become fragmented, which can happen when cast members of varying sizes are loaded and unloaded repeatedly. In this case, Director may *thrash* (repeatedly load and unload the needed cast members). Clear (and defragment) memory using *unLoadMember* as follows:

```
if the freeBlock < 200 * 1024 then
  unLoadMember
end if
```

The entry under *ramNeeded* in Macromedia's *Lingo Dictionary* includes an example that checks the *ramNeeded()* against *the freeBytes* to determine whether enough memory is available, but this is misleading. The *ramNeeded()* overstates the required memory if some cast members are already loaded (but understates the RAM needed if preloading digital video). Furthermore, *the freeBytes* understates the available memory, unless all purgeable cast members have been unloaded. To get accurate information, you must unload all cast members at the risk of having to reload some of them.

Reader Exercise: Write a *frameLoaded()* utility that checks whether the cast members needed for a frame are loaded. Model its syntax after the *ramNeeded()* and *frameReady()* functions. (Hint: use *the loaded of member* property to check each sprite's cast member. Use *go frame* to move the playback head so that you can check frames other than the current frame.) *FrameReady()* is not helpful because it indicates only whether items are available locally, not whether they are actually loaded.

The traceLoad

The traceLoad can diagnose loading problems by displaying in the Message window those cast members that are being loaded. It has three possible settings:

0: No output (the default). Loading is not shown.

1: Shows the cast member name or number of cast members being loaded.

In D6:

```
preLoadMember 200, 201
Loaded cast 200
Loaded cast Roulette Wheel
```

In D7:

```
member 200 of castLib 1("Roulette Wheel") was loaded into memory
```

2: In D6, shows the cast member name or number, current frame, current movie, and file seek offset of cast members being loaded, such as:

```
Loaded cast 200 frame=1  movie=Test seekOffset=-598
```

In D7, in addition to the information shown when *the traceLoad=1*, *the traceLoad=2* shows:

```
Time = 1470997 msec (237 msec later)
Movie "test" is on frame X (freeByes = 26616908, 49200 bytes
consumed)
Member is in movie (or external castLib) "filename"
File Seek Info: No file access occurred
File Seek Info: Seeked between files! Seeked 9014 to final
member at filepos 19262. Read in 4404 bytes.
```

Using *the traceLoad=2* will crash D7.0 unless you first save the movie and any castLibs. This bug is fixed in D7.0.1.

In D6's *traceLoad* output, the *Loaded cast* indicates only the cast member's *memberNum*, not its *castLibNum*. The *frame* is the current frame, not the frame of the Score for which the cast member was loaded (so it is useless when performing explicit loading via Lingo). The *movie* is the internal or external castLib's name. The *seekOffset* indicates the distance that Director had to seek (forward or backward) in the file to find the requested cast member. Large seek offsets imply that the cast members are stored inefficiently. Perform a `File ➤ Save and Compact` before using *the traceLoad* to diagnose file seek problems.

In D6, *the traceLoad*'s output does not show cast members being unloaded, nor does it show cast members that are already loaded. Set *the traceLogFile* to capture the output to an external text file for later analysis.

In D7, unloading is shown as:

```
member X of castLib Y ("Name") was purged from memory
```

Determining the compressed size on disk

In D7, use *the traceLoad=2* to analyze disk usage for cast members. In D6, you can't easily tell the size or relative positions of cast members in the Director movie file on disk. Example 9-1 uses the *seekOffset* from *the traceLoad* output in D6 to estimate the disk size of a cast member.

Example 9-1: Guessing an Asset's Size on Disk

```
on checkSeekOffset whichMember
  unLoadMember whichMember
  preLoadMember whichMember
  set the traceLoad = 2
  unload member whichMember
  preLoadMember whichMember
end
```

The result in the Message window will be something like:

```
    Loaded cast x  frame=1  movie=test  seekOffset=-76718
```

The first preload positions the disk head at the end of the cast member. The second preload forces Director to seek back to the beginning of the cast member. The absolute value of the *seekOffset* is the approximate size of the asset on disk.

Determining what is loaded

The utility in Example 9-2 reports which cast members are loaded and how much memory they occupy. You may want to save the Director movie and unload all cast members before running *showLoaded*.

Example 9-2: Determining Currently Loaded Cast Members

```
on showLoaded startCastLib, endCastLib
  if startCastLib = void then
    set startCastLib = 1
  end if
  if endCastLib = void then
    set endCastLib = the number of castLibs
  end if
  -- Track count and memory used for various member types
  set memberCount   = 0
  set memSize       = 0
  set scriptCount   = 0
  set scriptMemSize = 0
  set modCount      = 0
  set modMemSize    = 0
  set neverCount    = 0
  set neverMemSize  = 0
  -- Look in all castLibs for loaded cast members
  repeat with x = startCastLib to endCastLib
    repeat with y = 1 to the number of members of castLib x
      -- If a cast member is loaded, try to determine why
      if the loaded of member y of castLib x then
        set thisSize = the size of member y of castLib x
```

Example 9-2: Determining Currently Loaded Cast Members (continued)

```
        set memSize = memSize + thisSize
        set memberCount = memberCount + 1
        set thisType = the type of member y of castLib x
        case (thisType) of
          #script:
            -- Scripts are always loaded
            -- Don't bother printing them out
            set scriptCount = scriptCount + 1
            set scriptMemSize = scriptMemSize + thisSize
          otherwise:
            if the modified of member y of castLib x then
              -- Not unloadable because it has been
              -- modified since movie was last saved
              set append = "(MODIFIED)"
              set modMemSize = modMemSize + thisSize
              set modCount = modCount + 1
            else if the purgePriority of member y ¬
              of castLib x = 0 then
              -- Not unloadable because it is
              -- set to Never Unload
              set append = "(NEVER UNLOAD)"
              set neverMemSize = neverMemSize + thisSize
              set neverCount = neverCount + 1
            else
              -- Not unloaded because it is one of the other
              -- unloadable types (shapes, transitions, fonts)
              set append = EMPTY
            end if
            put member y of castLib x && "Type:" && ¬
                  thisType && append
        end case
      end if
    end repeat
  end repeat
  -- Display some statistics about what is still loaded
  put "Total size of" && memberCount && ¬
       "loaded member(s):" && memSize / 1024 && "KB"
  put "Including" && scriptCount && "script(s) totaling:" ¬
       && scriptMemSize / 1024 && "KB"
  put "Including" && modCount && "modified member(s)" && ¬
       "totaling:" && modMemSize / 1024 && "KB"
  put "Including" && neverCount && "unpurgeable" && ¬
       "member(s) totaling:" && neverMemSize / 1024 && "KB"
end showLoaded
```

Manual Preloading

Director allows you to attempt to preload cast members manually from disk into memory. Preloading stops when memory is full, so not all preload attempts succeed completely. Preloading cast members that are not ultimately needed is counterproductive, as is preloading additional cast members after memory is full

(items preloaded in the first batch would be unloaded to make room for the second batch). Manual loading of cast members works best when sufficient memory is available and the preloaded members will be used imminently.

Manual preloading does not appear to be working correctly in D7.0. Upgrade to D7.0.1, which remedies several preloading bugs.

Preloading large amounts of data can be time-consuming. Set the *preLoadEventAbort* to `TRUE` to allow the user to interrupt preloading, or set *the idleLoadMode* to use idle loading as described Table 9-11. The preload commands do not generally preload externally linked files. Digital video files are preloaded only if *the preLoad of member* is set. SWA files are preloaded according to *the preLoadTime of member* property using the *preLoadBuffer* command.

The preloading commands shown in Table 9-10 return a status in *the result* that is of dubious value in determining whether they succeeded. Entries preceded by "the" are properties, not commands.

Table 9-10: Preloading Commands

Command	**Usage**
downloadNetThing *URL*, *localFile*	Downloads data from *URL*, so that it is local when needed.
preLoad *toFrame*	In D6 and D7.0.1, preloads all cast members used in the Score from the current frame to frame *toFrame*. In D7.0, use *preload the frame, toFrame* instead.
preLoad *fromFrame*, *toFrame*	Preloads all cast members used in the Score from frame *fromFrame* to frame *toFrame*.
preLoad member *fromMember*, *toMember*	Preloads a range of cast members, as would the *preLoadMember* command.
the preLoad of member *videoMember*	Determines whether a *#digitalVideo* or *#QuickTimeMedia* cast member's external video file can be preloaded. See *the preLoadRAM* entry.
preLoadBuffer member *swaMember*	Preloads a portion of an SWA as set by *the preLoadTime of member*.
preLoadCast	Obsolete. See *preLoadMember*.
the preLoadEventAbort	If TRUE, allows a mouse click or key press to abort preloading. Defaults to FALSE.
preLoadMember	Preloads[1] all cast members in the current movie in D6. In D7, it preloads only the first castLib.

Table 9-10: Preloading Commands (continued)

Command	Usage
preLoadMember member *fromMember* {of castLib *fromCast*}, {member *toMember* of castLib *toCast*}	Preloads[1] a range of cast members, or a single cast member if *toMember* is omitted (*fromCast* and *toCast* can be different castLibs in D6, but preloading across multiple castLibs fails in D7).
the preLoadMode of castLib *whichCast*	Determines whether cast members are loaded when needed, or preloaded when the movie starts.
preLoadMovie *whichMovie*	Preloads a movie (URLs allowed) in anticipation of using *go movie* or *play movie*. *The result* returns 0 if successful or a negative error code.
preloadNetThing (*URL*)	Asynchronously preloads an HTML, GIF, or other web-based asset. Check the status with *netDone( )*.
the preLoadRAM	Specifies RAM (in bytes) used for preloading digital videos whose *preLoad of member* property is TRUE. When set to 0, *all* available RAM is used.
the preLoadTime of member *swaMember*	Determines the amount of sound preloaded (in seconds) when using *preLoadBuffer* or otherwise loading an SWA cast member.
the result	Used to access the result of a preload operation.

[1] *PreLoadMember* requires that *fromMember* and *toMember* be valid member names or non-empty member slots. *The number of members of castLib* property returns the last valid member slot, but that slot may be empty if cast members have been moved or deleted. The *number of castLibs* returns the number of the last castLib.

preLoadMember and preLoad

There may be insufficient memory to preload the cast members as requested via the *preLoad* and *preLoadMember* commands. Both return a status via *the result* that ostensibly can be used to check whether the operation succeeded. The following discussion applies mainly to D6. At press time, D7.0's preloading was too buggy to test effectively and D7.0.1 was not yet finalized.

preLoad and *preLoadMember* attempt to preload cast members in the order in which they are saved in the movie file or external castLib on disk, and not in *memberNum* order or the order in which they appear in the Score. Cast members are loaded from the lowest numbered castLib first.

The sequential cast member loading order is efficient when reading from disk, but leads to this oft-repeated potentially incorrect Lingo code:

```
preLoadMember 1, 5
put the result
-- 4
if the result < 5 then alert "Not enough memory to preload"
```

If cast members are used in the Score in the order 1, 2, 3, 5, 4, they will be stored to disk in the same order. *The result* will return 4 (the *last* cast member loaded), and not 5 (the *highest* numbered cast member loaded), implying an error, when in fact the *preLoadMember* command succeeded! (*The result* would return cast member 4, even if member 4 was already loaded, because *the result* indicates the

last cast member that would be loaded if needed. It does not necessarily return the highest number member loaded, nor the member that was actually loaded last.) To make *the result* return a meaningful answer, use `Modify ➤ Sort ➤` *Usage in Score* and then use `File ➤ Save and Compact` for internal casts. The sort order doesn't matter for members in external castLibs, because they are stored on disk in the order in which they appear in the Cast window.

Furthermore, if you preload frames 1 to 10, and the Score contains references to both internal and external cast members, members in the lowest number castLib are always loaded first. That is, if you need member 5 of castLib 2 in frame 1 of a movie, it may not be preloaded until after member 1 of castLib 1, even if the latter is not used until frame 2 of the movie! These issues may be more pronounced in Shockwave, where preloading can be slower.

If you attempt to preload items in smaller chunks, it is possible that those items preloaded with the first *preload* command will be unloaded by a subsequent *preload* command. In other words, preloading works best when you perform only one preload before using the newly loaded cast members. You can also unload all cast members before a series of preloads to increase the chance that your new cast members won't be unloaded.

The result of a *preLoadMember* command returns only the cast member's *memberNum*, leaving its *castLibNum* ambiguous:

```
preLoadMember member 2 of castLib 1
put the result
-- 2
preLoadMember member 2 of castLib 4
put the result
-- 2
```

Check *the loaded of member* property to confirm which cast members were loaded.

The *preLoad* command sets *the result* to the number of the last frame for which cast members were successfully preloaded, such as:

```
preLoad 1, 10
if the result <> 10 then alert "Couldn't finish preload"
```

Preloading digital video

A digital video's external data is not preloaded unless the *Enable Preload* option is set in its Cast Member Properties dialog box (or via *the preLoad of member* property). *The preLoadRAM* property determines the amount of memory used for preloading digital videos. The default setting (0) uses *all* available memory. Refer to Chapter 16, *Digital Video*, for more details on proper digital video preparation and optimizing digital video performance in Director.

If a digital video's data rate is sufficiently low, there should be no need to preload data, which should be provided on the fly. One exception might be a very small digital video that must play smoothly without dropping frames. To avoid accessing a CD-ROM in two places at once, preload animations or sounds instead, as they tend to be smaller than digital video.

The example for *the preLoadRAM* in Macromedia's *Lingo Dictionary* prior to D7 was meaningless. You should either specify a fixed value for preloading digital video, such as:

```
set the preLoadRAM = 2 * 1024 * 1024 -- 2 MB
```

or specify a percentage of *the freeBlock* for preloading, such as:

```
set the preLoadRAM = 0.5 * the freeBlock
```

The optimal setting would depend on the length of the video, the video's data rate, the free RAM, and the speed of the drive providing the data, but I prefer not to preload videos at all.

Asynchronous and Idle Loading

Typically, you might preload cast members for a segment in the Score and then wait a considerable time for the user to move onto the next screen. Director will generate one or more *idle* events during each frame as time allows. For example, if the Score's tempo is 10 fps, each frame is allotted six ticks. If Director takes only two ticks to draw the frame, 4 ticks would be left to process *idle* events. Director's idle loading commands (shown in Table 9-11) load cast members without monopolizing Director's attention.

Idle loading works in conjunction with the *preLoad* commands listed in Table 9-10. The frequency and duration of the attention paid to idle loading is controlled by the *idleLoadMode*, *idleHandlerPeriod*, *idleLoadPeriod*, and *idleReadChunkSize* properties.

The frequency of *mouseEnter*, *mouseLeave*, and *mouseWithin* events also depends on *the idleHandlerPeriod*. If *the idleHandlerPeriod* is not zero, rollover event handling becomes unusably sluggish.

Table 9-11: Idle Loading

Command	**Usage**
cancelIdleLoad *loadTag*	Aborts preloading of cast members with the specified *loadTag*.
finishIdleLoad *loadTag*	Finishes preload command previously tagged by *loadTag*.
on idle	The *on idle* handler is called during each idle event, but need not be used for idle loading.
the idleHandlerPeriod = *numTicks*	Increasing *the idleHandlerPeriod* reduces the frequency with which the *idle, mouseLeave, mouseEnter,* and *mouseWithin* events are generated. (Default is 0 in D6, which sends these events as frequently as possible and 1 in D7, which reduces Director's monopolization of the processor.)
idleLoadDone(*loadTag*)	Returns TRUE if cast members with specified *loadTag* are all loaded.
the idleLoadMode	Optionally allows idle loading (default is *Never*). See the next section.
the idleLoadPeriod = *numTicks*	Increasing *the idleLoadPeriod* increases the time between idle loads, allowing more time for idle activities other than cast member loading.

Table 9-11: Idle Loading (continued)

Command	Usage
the idleLoadTag = *loadTag*	Creates an arbitrary *loadTag* that identifies the batch of cast members loaded by the next preload command.
the idleReadChunkSize = *chunkSize*	Size (in bytes) of data read each time idle load queue is serviced. Defaults to 32,767 bytes (32 KB).

The idleLoadMode

The idleLoadMode has four possible values:

0: Cast members are immediately preloaded when a *preLoad* command is issued. (No idle loading. This is the default.)

1: Performs idle loading when there is free time between the *enterFrame* and *exitFrame* messages in a frame.

2: Performs idle loading each time Director calls the *on idle* handler. The *idleHandlerPeriod* limits how often *idle* events will be sent (the default is as frequently as possible) and *the idleLoadPeriod* determines whether Director will attempt multiple idle reads during an idle event (the default is to load as frequently as possible during an idle event).

3: Performs idle loading as frequently as possible (both while idling in a frame, and again after exiting a frame before reaching the next frame).

Each time the idle load queue is serviced, Director reads data equal in size to *the idleReadChunkSize* (defaults to 32 KB).

If you expect the user to follow a particular path, you might preload the cast members used in a range of frames. Example 9-3 shows how to initiate and then wait for idle loading.

Example 9-3: Idle Loading

```
on exitFrame
  -- Idle load as frequently as possible
  set the idleLoadMode = 3
  -- Identify this idle load batch with an arbitrary number
  set the idleLoadTag = 1
  -- Idle load the next scene in the Score
  preLoad marker(1), marker(2)
end
```

Then, in a subsequent frame, you might wait for idle loading to complete before jumping to the new marker:

```
on exitFrame
  if idleLoadDone(the idleLoadTag) then
    -- Turn off idle loading.
    set the idleLoadMode = 0
    -- Go to next marker if all cast members are loaded.
    go marker (1)
```

```
  else if the mouseDown then
    -- Finish idle loading if the user clicked the mouse.
    finishIdleLoad (the idleLoadTag)
    set the idleLoadMode = 0
    go marker(1)
  else
    go the frame
  end if
end
```

Note that we left *the idleHandlerPeriod*, *the idleLoadPeriod*, and *the idleReadChunkSize* at their default values, causing idle reading to occur in 32 KB chunks as frequently as possible.

Memory Optimization

The following sections help you avoid and diagnose memory problems.

Before you can fix a memory problem, you must confirm that the problem is memory-related. The following conditions may indicate a lack of available memory but may also have other causes:

- Warning errors from Director that it has used all available memory.
- Audio not being played or dropping out.
- Background graphics and large graphics not appearing on the Stage.
- Excessive disk access (thrashing). In very low RAM, Director may purge and reload the same graphic repeatedly.
- Sprites leaving trails even when *the trails of member* is `FALSE`.
- Flickering cursors.
- Printing is extremely slow or delayed until Director quits.
- Poor performance seen only on machines with less installed RAM.

Causes of Memory Errors

There are two types of so-called *memory errors*: either Director is running low on memory due to a known limitation, or there is an unintentional *memory leak* in one or more system components. Director may issue the error, "Warning! The current movie has used all of Director's main memory and some of its reserve memory," or even crash. Save your file immediately if you receive this error.

Not enough memory

You can allocate more memory to Director as described under "Memory Allocations" earlier in this chapter. The following will often lead to low memory situations:

Importing or creating too many cast members during authoring or at runtime
: Allocate more memory to Director, import fewer items, and resave the Director file to free memory. Avoid importing during runtime, or link to external files using *the fileName of member* property instead.

Cast members can not be unloaded

Save the file to allow cast members to be purged. Don't set *the purgePriority of member* to 0 (*Never*). Use *Normal*, *Next*, or *Last* settings instead.

Director is trying to load a nonexistent cast member

If Director gives the error, "Not Enough memory to load some cast members," it may be trying to load a nonexistent cast member, caused by an erroneous reference in the Score to a cast member that has been deleted. See *the traceLoad* later in this chapter and Example 3-9 to diagnose the problem.

External applications are not leaving enough RAM available to Director

Quit other applications before starting Director or a Projector, and check *the memorySize* and *the freeBlock*.

Director does not leave external applications enough memory to be launched using the Lingo open command

Use zLaunch (*http://www.zeusprod.com/products/zlaunch.html*) to quit the Projector while an external application runs and relaunch your Projector when the external application terminates.

Recursion

Recursion occurs when a handler calls itself or two routines call each other, either directly or indirectly. It often leads to a crash and should be avoided, unless it is intentional. In Example 9-4, the two functions call each other until Director runs out of memory due to an overflowed *handler call stack*. Set breakpoints in each handler to watch the call stack grow in the upper-left pane of the Debugger window (save your work first, as this may crash your computer).

Example 9-4: Bad Mojo Recursion

```
on handlerA
  handlerB()
end

on handlerB
  handlerA()
end
```

Memory leaks

The following are possible causes of memory leaks. Small leaks may never accumulate to the point where they cause trouble. You many need to test for a long time on a machine with little RAM to duplicate the problem. Not all of these things will always cause leaks. but if you suspect a memory leak, you should check these items first:

Using Lingo improperly or not freeing objects

An apparent memory leak may be caused by instantiating an Xtra or child object, creating a large list, or opening a MIAW but never disposing of it. You must eliminate references to objects to allow Director to perform "garbage collection."

A bug in or improper use of an Xtra or XObject

Try to isolate the cause of the problem and contact the Xtra manufacturer for instructions on proper use of the Xtra or to identify known bugs. Leaks from Xtras or XObjects are not uncommon, because C programmers must allocate and deallocate memory explicitly. Multiple layers, such as the ActiveX Xtra interfacing with an ActiveX control, increase the potential sources of error.

Cursor and menu leaks

Switching cursors in D6.0 caused a memory leak (although fixed in D6.0.1, it reappeared in D7.0, but should be lessened in D7.0.1). Switching menus repeatedly under Windows caused a severe memory leak in D5, and it is still recommended that you not switch menus or cursors excessively.

External castLib leaks

There have been reports of Sprite Xtras used in external Casts causing memory leaks. When in doubt, internal Casts are least likely to cause problems.

Sounds not being unloaded

Playing a puppet sound from an external castLib as you go to a new Director movie may cause the sound to stay in memory and never be unloaded (see *http://www.updatestage.com/buglist.html*). Unpuppet any sounds (using *puppetSound 0*) before going to a new movie.

Using play movie without play done

When using the *play movie* command, the first movie is kept in memory until you return from the second movie using *play done*. Avoid the *play movie* command if you can't guarantee that a matching *play done* command will eventually be issued. Use *go movie* and a global variable to store the name of movie to return to instead.

Creating or importing assets dynamically at runtime

Any items created dynamically must reside in RAM. During authoring, these can be purged only when the Director file is saved. From a Projector, they can cause memory to run low. Avoid dynamic linking of content or set *the fileName of member* property instead.

External applications

Anything that uses an external application relies on that application to behave properly. If that external application leaks memory, you won't find the error within your Director project.

External files

When using external files, such as with the FileIO Xtra, close any files when done to allow the OS to deallocate the file handle assigned to the file.

Isolating and Diagnosing Memory Problems

Director performs memory cleanup ("garbage collection") periodically. There is a lag between when Director *could* reclaim memory and when it actually does reclaim it. Decreased available memory does not necessarily indicate a memory leak. Director won't unload cast members unless it needs the memory for something else or until you use *unload* or *unLoadMember*. *The freeBytes* and *the freeBlock* typically drop very low, and then fluctuate at low levels as Director

unloads just enough cast members to make room for those it is about to load. In D7, *the freeBytes* shouldn't drop below 300 KB or so.

If memory decreases slowly over time, it may be the cumulative effect of a small memory leak. If memory drops suddenly and can't be reclaimed, it may be due to a large cast member not being freed from memory.

When you suspect a memory leak, try disabling any Lingo that performs any unusual and/or very repetitive operation. For example, if a project using an untested Xtra or unsupported Lingo command is leaking memory, disable those items first. Other suspicious items include dynamically editing properties at runtime that are ordinarily only changed during authoring. This would include changing member properties (as opposed to sprite properties), creating new cast members, Score recording, dynamically linking to external castLibs, and the like.

Use Alex Zavatone's MemMon utility to help diagnose memory problems. For the shareware version, search for "MemMon" at: *http://www.director-online.com/help_central/DOUGsearch/searches/DWdemo.html*. For technical and ordering information for the professional version, see *http://www.blacktop.com/zav/toolkit*. For Zav's article on memory management, see *http://www.director-online.com/howTo/UD_articles/UD26.html*.

Checking cast member loading and unloading

To display cast member loading, set *the traceLoad* in the Message window and play the movie:

```
set the traceLoad = 1
```

If a cast member reference is incorrect, you will see Director repeatedly try to load the problematic cast member that it can't find.

Check for *purgePriority of member* settings other than 3 (*Normal*) or 2 (*Next*) using a loop similar to that shown in Example 4-8. Use *the loaded of member* property to determine which cast members are currently loaded (see Example 9-2).

Checking memory usage

Director has several commands to check memory usage. You should save your file, then select an empty Score frame, and use *unLoadMember* or the *Purge* button in the Memory Inspector to clear memory. Use *put the freeBytes* and *put the memorySize* to establish a baseline for the available memory.

Create an *on idle* handler that displays the available memory in the Message window or in field cast members you've placed on the Stage (see Example 9-5).

Example 9-5: Idle Handler Displaying Memory Status

```
on idle
  put the freeBlock into field "FreeBlock Display"
  put the freeBytes into field "FreeBytes Display"
  put the memorySize into field "MemorySize Display"
end
```

At any time you should be able to recover some or all of the memory used by selecting an empty Score frame and using *unLoadMember*. If memory continues to decline, there may be a leak.

If writing to the Message window to diagnose a memory leak, be aware that each character displayed in the Message window consumes one byte of memory itself!

The diagnostic text printed in the Message window is purged when it exceeds 32 KB, but can appear to indicate a small memory leak when none exists.

Reducing Memory Requirements

Before allocating more memory, try to reduce the memory and bandwidth requirements for your project. Note that many of the following techniques improve performance as the memory use is reduced, which isn't surprising. On the other hand, using bitmaps instead of Flash vector sprites improves performance at the expense of memory usage. Options to reduce memory and bandwidth include:

- Use 8-bit (256-color) graphics with a custom palette instead of higher color depths.
- Use a smaller Stage size.
- Reduce the size of animations and the bitmaps used in them.
- Use fields or D7 text cast members instead of D6 rich text cast members.
- Stream large sounds from disk by linking to them externally, so that they don't remain in RAM.
- Reduce the sampling rate, bit rate, or number of channels for your audio.
- Pre-mix sounds rather than using multiple sound channels.
- Reduce the number of scripts by generalizing handlers or using Behaviors.
- Use shapes, especially shapes filled with tile patterns, rather than bitmaps.
- Never set *the purgePriority of member* to 0 (*Never*) and avoid setting it to 1 (*Last*).
- Don't import external files (using *importFileInto*) or create cast members using *new(member)* at runtime.
- Use *go movie* instead of *play movie* (the latter leaves the first movie partially in RAM).
- Avoid preloading large digital video cast members using *the preLoadRAM* and *the preLoad of member*.
- Set *the preLoadMode* of castLib to 0 (*Load When Needed*) for each castLib to improve startup times.
- Preload only cast members that you know will be needed.
- Reduce the number of MIAWs in use.

Performance

Many developers discover that their titles that ran fine from the hard drive perform poorly when they run from a CD-ROM or on a slower machine. You should have a minimum-capability test box available to you in all circumstances. Refer to Chapter 7, *Cross-Platform and OS Dependencies*, for additional details on how the machine configuration can affect performance and intonations of the "Test Early, Test Often" mantra.

See "Determining the Appropriate Minimum Hardware Playback Platform" at *http://www.macromedia.com/support/director/how/expert/playback/playback.html*, and also *http://www.zeusprod.com/technote/machspec.html*. These articles discuss how to target the appropriate segment of the installed base to meet both your technical and marketing requirements.

Most developers create a project ad hoc and then see if it runs. You should instead design your project with the optimization principles discussed throughout this book in mind. This will reduce the likelihood and severity of any problems.

Often, a customer will report that your beta version does not run satisfactorily on his machine. Ideally, you can test it yourself on the customer's machine(s) in his presence. If this is impossible (and not just impractical) you must get the customer to tell you precisely which portions are unsatisfactory and the characteristics of the test machine. It is nearly impossible to diagnose and debug a problem that you can't replicate, and you won't know whether you've solved it until the client performs another round of testing. This is egregiously inefficient. Your goal should be to minimize the retesting cycle by installing Director on the machine(s) demonstrating the problem. If you don't see the problem on your test machines, obtain a machine with a comparable—preferably identical—configuration.

If necessary, you can simulate a crippled machine by removing RAM, running a CD-ROM over a network, running other applications at the same time, or using much larger versions of the media (such as 44 kHz, 16-bit, stereo sound) and playing back at higher color depths. You can also limit the amount of RAM in use by a Macintosh Projector or by Director during authoring as described earlier in this chapter. This may place enough of a load on Director that you can see the problems the client describes.

Refer to Chapters 12 through 16 to ensure that you've prepared your content optimally, as improper content preparation often contributes to poor performance.

Macromedia TechNote #08151, "Why does the NT CPU monitor go to 100% when I run Director?", provides excellent details on Director's requests for CPU time under Windows. Director soaks up all available *idle* events. This avoids potentially jerky animation caused by infrequent attention. Director only appears to consume 100% of the processor, because the Windows NT CPU meter measures idle event usage only by applications, not the system. Director does not prevent other applications from receiving time slices when needed. See the D7 *ReadMe* for more information.

Gauging Performance

You can measure the speed of a given operation by starting a timer with *startTimer,* and then checking *the timer* after the operation completes. Example 9-6 checks the speed of 1,000 executions of the *offset()* function. It could be used to compare processor performance or determine whether a particular Lingo command is relatively slow.

Example 9-6: Gauging Performance

```
on testSpeed
  startTimer
  repeat with i = 1 to 1000
    -- Test the offset() function
    set dummy = offset ("st", "test")
  end repeat
  put "Test took" && the timer && "ticks"
end testSpeed
```

Don't include a *put* statement within your timing loop because printing to the Message window is very slow. Store the result in a variable and print it after the timing test completes.

You can also start a timer in one frame, check it in a later frame, and then divide by the number of frames to calculate the approximate frame rate achieved when the movie runs.

Things That Hurt Performance

The following sections list optimization techniques. In some cases, you may need to specify a faster minimum playback platform with more memory and a faster CD-ROM or Internet connection.

Poor performance is often due to a confluence of factors or the cumulative weight of techniques that may not compromise performance in isolation. For example, if you are using SWA sounds, streaming high bandwidth video, and loading large graphics all at once, the cumulative load will overpower low-end machines. Using a high color depth and sprites with alpha channels adds insult to injury. Likewise, eliminating a single culprit may not improve performance measurably, but the cumulative benefit of small changes can be great.

Do not expect optimal performance from versions of Director that predated a given operating system. For example, you should consider upgrading to D7 if you are supporting Windows 98. If you encounter performance problems, perform tests with the latest version of Director.

Lingo scripting techniques or inherent Lingo sluggishness

Following are some Lingo performance tips. Some improve your Lingo code's readability; others compromise it for maximum performance (include comments in your Lingo as necessary):

- Alex Zavatone benchmarked some Lingo variable allocations and simple operations. His Lingo profiler is for sale too. See *http://www.blacktop.com/zav/perf.txt*.
- Avoid tight repeat loops, which lock out other processing, especially in Shockwave.
- Avoid lengthy *on idle* handlers. They can slow performance, because they may be called several times during each frame.
- Printing to the Message windows is slow. Turn off *the traceLoad* and *the trace* commands in the Message window and eliminate extraneous *put* statements.
- Updating fields on Stage is slow, especially if done very frequently.
- Text parsing and searching is slow for long strings. Use the Text Cruncher Xtra (*http://www.itp.tsoa.nyu.edu/~student/yair/texcruncher/HTML/YairText-Cruncher.html*) to parse large amounts of text or use sorted lists for searching.
- String comparisons, especially with long strings and fields, are slow. Use symbols (or integers) instead of strings when possible.
- Leave *the romanLingo* = `TRUE` (the default for most languages) for faster performance, unless using Japanese or another double-byte language.
- Accessing cast members by number is faster than by name, but referring to them by name is easier, should their cast member number change. Director caches cast member names, so that accessing by name should be fast the second time, but it does not cache script names.
- Accessing a frame by number is faster than accessing it by its marker label name, but referring to it by name is easier, should its frame number change. If you're accessing a frame frequently, assign the number returned by *label("frameName")* to a global variable for future use.
- Change *the cpuHogTicks* and *the idleHandlerPeriod* with caution. Despite accelerating some operations, they can affect others adversely.
- Minimize use of *the actorList* and indiscriminate broadcasting of messages using *sendAllSprites()*.
- The *value()* function and *do* command invoke the Lingo parser to evaluate a string expression, which can be quite slow. For example, use *duplicate()* instead of *value(string())* to duplicate a list.
- List operations tend to be fast. Sorted lists are faster than unsorted lists when accessing elements by value or property name, but not when accessing elements by their index (position).
- Unnecessarily converting between data types, such as converting strings to numbers, can be slow.
- Reduce the number of properties being set. Setting *the rect of sprite* is faster than setting the *width* and *height of sprite* separately. Setting *the loc of sprite* is faster than setting the *locH* and *locV of sprite* separately.

- Reduce the number of calculations by moving static calculations outside repeat loops. Use a global variable rather than calculating a number in a handler that is called repeatedly. Precalculate static numbers. For example:

  ```
  set a = b * 180/pi
  ```

  is not as fast as:

  ```
  set a = b * 57.296
  ```

- Floating-point calculations take from 1.5 to 3 times longer than integer calculations, with multiplication and division being the slowest. Use fixed-point math whenever possible, but remember that *the maxInteger* is smaller than the maximum floating-point number.
- Eliminate redundant *updateStage* commands that unnecessarily refresh the screen. It refreshes automatically once per frame as the playback head moves or loops.
- Comments have no effect on performance, but the *nothing* command takes a small (but measurable) time to execute. There is no speed difference between D7's new dot notation and the older Lingo syntax.

Media and disk access

There are dozens of reasons why media or disk access might go awry. Here are some common problems:

- Avoid using *the searchPath* (or *the searchPaths*), especially with a long list of paths to search. Linking to external files using an explicit path is much faster.
- Avoid streaming media from two or more external files on the same CD-ROM simultaneously. Preload some data into memory or stream it from the user's hard drive.
- Avoid more than four concurrent Internet streaming operations.
- Stream large sounds from disk by linking to them externally so that they don't need to be fully loaded into RAM before they start playing.
- Avoid playing two or more sounds simultaneously under Windows (mixing sounds causes an initial delay).
- Avoid linked external bitmaps, which are slow to decompress. Import them into the cast instead.
- Don't save in Shockwave (compressed) format for local content, because Shocked data takes longer to decompress.
- Optimize placement of files on a CD-ROM or defragment your hard drive.
- If the CD or hard disk's access light is constantly flickering, it indicates so-called *thrashing*, in which data is constantly being loaded and unloaded. The disk swap file space is much slower than real RAM.
- File access using the FileIO Xtra can be slow for large databases. Use a third-party database Xtra.
- Turn off networking and disable extraneous extensions.

- Too many Xtras or non-Xtra files in the *Xtras* folder will slow down Director or a Projector's startup. It also makes file saves slow during authoring.
- A corrupted Score can cause Director to thrash in an attempt to load a non-existent cast member. See Example 3-9 to detect a corrupted Score.

Digital video

Digital video performance depends primarily on the proper creation of the digital video file in your digital video editing software and proper compression. Some tips to eliminate performance-robbing culprits:

- Incorrect interleaving can be devastating to performance.
- Reduce the data rate to a bandwidth appropriate for the minimum CD-ROM drive speed.
- Play video direct-to-Stage using *Sync to Soundtrack* mode.
- Place digital video sprites on four-pixel boundaries on the Stage.
- Stretch digital video sprites in increments of 100% only.
- Use digital video instead of animation when timing is critical (digital video will drop video frames if necessary).
- Waiting for digital video via Lingo can be more efficient than waiting via the Tempo channel.

Sound

Tips on sound optimization and reducing latency (see also Chapter 15):

- Use a lower sampling rate, bits per sample, or fewer channels.
- SWA uses less disk space and download time, but requires more processing power.
- Use only one sound channel under Windows.
- Use the same sampling rate and bit depth for all sounds played simultaneously. Use the standard sampling rates (11.025, 22.050, and 44.100 kHz).
- Use *updateStage* to trigger *puppetSound* commands.
- Stream large sounds from disk.

Score, Cast, and window usage

There are a number of non-intuitive quirks to Director that can degrade performance:

- Don't loop in a frame with a transition or use a transition while a digital video is playing (especially under Windows).
- Don't loop in the first frame or last frame of the Score. These frames incur an overhead penalty.

- Reduce the number of sprite channels and film loops in use. Set *the lastChannel* in D7 under `Modify ➤ Movie ➤ Properties` to some number less than the default (150).
- Break up movies with extremely large Casts or Scores into multiple movies.
- Eliminate unneeded cast libraries. An excessive number of linked external castLibs (more than 6 or so) consumes excessive RAM and degrades performance.
- Use a realistic frame rate (10 to 20 fps) for your minimum target platform.
- Reduce the number of animations, sounds, and digital videos playing simultaneously.
- Delete unneeded cast members and perform a `File ➤ Save and Compact` to optimize the Cast on disk.
- Close any window that refers to data that is being changed at runtime, including the Cast, Score, Control Panel, Message, and Watcher windows (and remove items from the list of watched expressions).
- Cast members in external castLibs are not stored in the most efficient order unless you use `Modify ➤ Sort` to sort them in the order used in the Score.

To check whether a sprite or Effects channel is degrading performance, use the *Mute* button in the Score to disable that channel (including sound channels) temporarily.

Graphics and animation

Graphics and animation are at the heart of Director, and a common performance bottleneck:

- The DirMMX Xtra will improve certain graphics performance in D6 on MMX-capable Windows machines.
- Set all monitors to the same color depth, preferably 8-bit (256 colors), and use graphics with the same bit depth as the monitor.
- Reduce and optimize animations.
- Avoid stretched bitmaps. Leave sprites at their cast member's *native* size or use `Modify ➤ Transform Bitmap` if necessary.
- Avoid slower inks, especially Blend. Use Copy and Background Transparent inks. Mask and Matte inks use twice the memory of any other ink, because Director must create a duplicate of the artwork internally. The allocation required for Matte ink makes it slightly slower the first time it is used for a sprite.
- Setting trails for static images can reduce the number of sprites needed.

- Turn off *the antiAlias of member* or adjust *the antiAliasThreshold* of text members to speed rendering. See also D7.0.1's *preRender* and *saveBitmap* properties in Table 12-8.
- Turn off *the dither* and *the useAlpha of member* for bitmaps when possible.
- When you have a choice, play assets such as Flash and QuickTime direct-to-Stage.
- Improve perceived performance by providing user feedback such as a wait cursor or button feedback.
- QuickDraw shapes use less memory but draw more slowly than bitmaps. Vector shapes and Flash cast members are the slowest to draw, and the speed of drawing depends on their scale to an extent.
- Animations performed via Lingo can be much faster than animations in the Score. Use Lingo to cycle through a series of cast members in a list or in adjacent slots in the Score, using *updateStage* each time you change *the member of sprite* property to display the new cast member.
- Avoid 32-bit graphics.
- Avoid cast members larger than the Stage.
- Set *the useFastQuads* to `TRUE` to improve performance when using *the quad of sprite*.

Lingo Affecting Performance

There are several commands that affect how Director allocates CPU time. Refer to Chapter 11 for details on asynchronous operations (such as waiting for media to be downloaded from the Internet).

The cpuHogTicks

The cpuHogTicks affects how often Director for Macintosh allows other processes to obtain the processor's attention. It has no effect under Windows, and has been available (although undocumented) since Director 4. The default value (20) allows other processes to interrupt Director every 20 ticks (three times per second). It can be increased (judiciously) to avoid releasing CPU control in the middle of an operation. It will not increase performance as much as it will prevent an operation from being interrupted, and therefore provides smooth animation without an intermittent hitch.

The marginal performance benefit from increasing *the cpuHogTicks* is no substitute for proper optimization. It can also interfere with mouse, clock, keyboard, network, and other system events.

Set *the cpuHogTicks* to 0 to speed auto-repeating *keyDown* events while holding down a key.

The idleHandlerPeriod

The idleHandlerPeriod determines how often Director allows idle events to be processed. If *the idleHandlerPeriod* is increased from the default (0 in D6, and 1 in

D7) idle events are processed *less* frequently. Set *the idleHandlerPeriod* to 0 to allow Director to monopolize the processor. Anything that depends on idle events, including the *on idle* handler, idle loading, and *mouseEnter*, *mouseLeave*, and *mouseWithin* events can become unusably sluggish at higher values.

The netThrottleTicks

The netThrottleTicks affects how often Director for Macintosh allows network operations to be interrupted. It is officially supported in D7, but was undocumented in D6. It has no effect under Windows or in Shockwave. The default value is 15 ticks (which allows net operations to be interrupted four times per second). Lowering *the netThrottleTicks* causes Internet-based media to download somewhat faster. Increasing *the netThrottleTicks* gives priority to local operations such as animation.

The fullColorPermit

The fullColorPermit determines the color depth of the offscreen buffer. To improve performance, set it to `FALSE` when using 8-bit graphics on monitors with a higher color depth. It is obsolete in D7.

The lastChannel

D7 allows up to 1000 sprite channels, but this degrades performance. Set *the lastChannel* to the smallest number of sprite channels you need under `Modify` ➤ `Movie` ➤ `Properties`. It is not settable via Lingo.

The useFastQuads

This undocumented D7 property renders sprites distorted with the new *quad of sprite* property more quickly but with lower quality. The default is `FALSE`. Setting it to `TRUE` also remedies some display bugs in D7.0.

The Big Squeeze (Fitting Your Project on a Floppy)

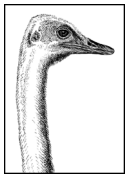
You'll often want to distribute a portfolio or demonstration piece on a floppy. That is becoming increasingly unrealistic as well as unnecessary. The compromises and time required to fit things on a floppy are rarely warranted these days. To wit:

- CD-R blank disks are very inexpensive (about $1) and CD-ROM burners are under $300. All your clients will have CD-ROM drives.
- Many clients have Zip drives; Zip disks are about $10 apiece (and hold 100 MB or more).

If space is at a premium:

- Use D7's new Slim Projectors (about 200 KB). These require that the user has Shockwave 7 installed or be willing to download it when first launching the Slim Projector.
- Use an older version of Director that creates smaller Projectors (see Table 8-1). Use Windows 3.1 and Mac 68K Projectors, which are smaller than Windows 95/98/NT, PowerMac, or FAT Projectors.

- Compress your files using StuffIt!, WinZip, or a similar utility. Most compression utilities can create self-extracting executable archives. They can also split an archive over multiple floppies.
- Submit a Shockwave version of the file that the user can play in a browser or using ShockMachine, or post it to a URL and let the user play it over the Internet.
- Use external graphics in JPEG or GIF format (the Xtras required for these may undermine any space savings in D6). D7 DCR and CCT files support internal JPEG and GIF compression.
- Use SWA compression for both internal and external sounds.
- Use lower quality graphics, video, or audio, which should require less storage. Crop images, use lower color depths, and compress video more aggressively.
- Use an empty *FONTMAP.TXT* file to save a few kilobytes.
- Omit unnecessary Xtras.
- Use DCR and CCT files, which are the smallest, or DXR and CXT files, which are slightly smaller than DIR and CST files.
- Always perform a `File➤Save and Compact` to purge deleted cast members.



CHAPTER 10

Using Xtras

Xtras allow Macromedia and third-party developers to extend Director's core capabilities and are analogous to the plug-ins available for many software products. This chapter covers the selection, installation, and use of Xtras in D5 through D7, with a focus on non-Lingo Xtras. Lingo Scripting Xtras, which replace the older XObjects supported in D3.1.3 through D6, are covered in Chapter 13, *Lingo Xtras and XObjects*, in *Lingo in a Nutshell*.

Xtras allow Macromedia to update individual components of Director or Shockwave without a major release. Director 7 relies more heavily on Xtras than any previous version. Xtras are well integrated into Director. You may not be able to distinguish a built-in feature from one that uses an Xtra. Even if you don't use Lingo, you can use many non-Lingo Xtras. Some developers want to use an Xtra for everything; some want to avoid Xtras at all costs. Neither extreme is justified. See "Do You Need an Xtra?" later in this chapter.

See *http://www.zeusprod.com/nutshell/xtras.html* for the latest information on D7 Xtras, including Xtras packaging and automatic downloading in Shockwave.

Types of Xtras

There are several distinct types of Xtras supported by Director. Some "Xtras" are actually a combination of more than one Xtra type; a Sprite Xtra may include a companion Lingo Xtra that allows you to manipulate the new Sprite type via Lingo. Xtras that create new custom cast member types (Sprite Xtras or Transition Xtras) are called Asset Xtras. Some Xtras are used during authoring only, but many must be included with your Projector to support specific features. Types of Xtras include:

Lingo Xtras or Scripting Xtras
: Lingo Xtras add new commands to Lingo, such as the ability to read and write external files (provided by the FileIO Xtra). Lingo Xtras are a replacement for older XObjects supported in prior versions of Director (but not in D7). You can also "extend" Director by writing Lingo handlers, but Lingo Xtras are typically written in C/C++.

Sprite Xtras or Asset Xtras

Sprite Xtras add new cast member types, such as QuickTime 3, Flash, ActiveX, or Custom Cursor cast members to the built-in types (bitmaps, fields, and so on). Sprite Xtra cast members can be placed on the Stage like any other sprite. The developer of the Sprite Xtra (Macromedia or a third party) determines its attributes, such as whether it has custom properties, supports a media editor, is imaged direct-to-Stage, and supports ink effects. Macromedia ships two versions of its Sprite Xtras—one for authoring and one for distribution with the Projector.

Transition Xtras

Transition Xtras appear alongside the built-in transitions (which don't require Xtras) in the Transition dialog box. Transition Xtras create custom transition cast members, and are a type of Asset Xtra. The developer of a Transition Xtra determines whether it supports the change area, chunk size, and duration options common to most transitions, and whether it can be interleaved with palette changes.

Tool Xtras

Tool Xtras are made available during authoring, usually via a windoid, and often analyze or modify the Cast or Score. The Animation Wizard (obsolete in D7) is written in Lingo, but most Tool Xtras are written in C/C++.

MIX Xtras (introduced in D6)

Media Information eXchange (MIX) Xtras import and export various graphic and sound formats, such as PICTs. They are required during authoring to import external media via drag-and-drop or `File ➤ Import` (and must be included with your Projector when using linked graphics and sounds). All MIX Xtras require the MIX Services Xtra.

Photoshop filters

Director can use some Photoshop 3 filters to modify bitmaps in the Paint window during authoring only. Photoshop 4.0 and 5.0 filters will not work. Refer to the *Filter Bitmap*, *Auto Filter*, and *Auto Distort* options under the `Xtras` menu. See also Chapter 13, *Graphics, Color, and Palettes*.

Non-Director Xtras

Other Macromedia applications support Xtras, although not necessarily the same ones as Director (some Xtras may support *multiple* Macromedia products). You'll need the SWA Export Xtra for SoundEdit or Peak LE to create Shockwave audio on the Macintosh. (SoundEdit Xtras go in the *System Folder:Macromedia:Xtras* folder and Peak LE Xtras go in the *Peak Plug-ins* folder.)

Sound Mixer Xtras (introduced in D7)

D7 implements Windows sound mixers as Xtras. MacroMix is contained within MacroMix.X32 and QT3Mix is contained within QT3Asset.X32. A DirectSound mixer, DirectSound.X32, was added in D7.0.1. See Chapter 15.

Miscellaneous and third-party Xtras

Some Xtras, including Shockwave Audio, PowerPoint Import, and Java Export, may not fit neatly in another category and show up in the `File` or `Xtras` menus. Others are used by Director transparently without showing up in any menu.

Xtras in the Interface

Macromedia places Xtras in subfolders somewhat arbitrarily. Regardless of the subfolder in which an Xtra is installed, the Xtra's internal type (and subtype) determines where it shows up (if at all) in Director's interface. Table 10-1 shows where you can expect an Xtra to appear once it is installed.

Table 10-1: Accessing Installed Xtras

Xtra Type	**Accessed Via**
Lingo Xtras[1]	Lingo only. See *showXlib*, *the xtraList, interface()*, and *mMessageList()* and Chapter 13 in *Lingo in a Nutshell*.
Sprite Xtras[1]	Insert ➤ Media Element, Insert ➤ Control, or File ➤ Import.
Transition Xtras[1]	Transition dialog box along with built-in transition types. See Modify ➤ Frame ➤ Transition.
MIX Xtras[1]	File ➤ Import and *importFileInto*. See also File ➤ Preferences ➤ Editors.
Export and Import Xtras	The Save as Java command appears under the File menu and requires the Java Export Xtras. The *Import PowerPoint File* option appears under the Xtras menu and requires the Import Xtra for PowerPoint. Other Export Xtras are used under File ➤ Export and are for authoring only.
Tool Xtras	Xtras menu (authoring mode only).
Shockwave Audio[1]	Insert ➤ Media Element, File ➤ Import (in D7), and Xtras menus.
Photoshop Filters	Xtras ➤ Filter Bitmap and Xtras ➤ Auto Filter.
Sound Mixer Xtras[1]	the soundDevice, the soundDeviceList

[1] Can be added to *the movieXtraList* under Modify ➤ Movie ➤ Xtras.

The Xtras menu

The `Xtras` menu includes items that are only marginally related (in that they use Xtras) and are available only during authoring, not from runtime Projectors. You can reorganize some items in the `Xtras` menu by changing the file structure within the *Xtras* folder (see following description). The first four options are built into Director and don't require Xtras:

Update Movies

D6 updates movies from D4 or D5. D7 updates movies from D5 and D6 only. After updating from D4 or D5, convert ranges of cells into sprite spans by selecting the cells, choosing `Modify ➤ Join Sprite`, then choosing `Insert ➤ Remove Keyframe`.

Update Movies also protects movies and castLibs (creates DXR and CXR files) and compresses movies and castLibs for Shockwave or local use (creates DCR and CCR files). In D5, use the AfterBurner Xtra. In D4, use the AfterBurner standalone executable (see Chapter 11, *Shockwave and the Internet*). See also the `File ➤ Save As Shockwave` menu option.

In each case, it allows you to make a backup copy of the original files. See Tables 4-1 and 4-3 for additional details.

Always keep your original source files. Never use the *Delete Original Files* option under `Xtras` ➤ *Update Movies*.

Filter Bitmap, Auto Filter, and Auto Distort

These options allow you to apply Photoshop 3.0 filters or one of Director's built-in transformations to cast members in the Paint window.

Widgets, buttons, Behaviors, wizards, and palette libraries (D6 only)

The Widget Wizard and Button Libraries allow you to add premade Lingo components to your project, such as fancy buttons with existing Behaviors. The Behavior Library lets you add Behaviors to sprites and frames, and the Animation Wizard automates rich text animation for speaker support. The Palette Library includes palettes with reserved colors in the first and last ten palette positions for use under Windows. In D7, the `Window` ➤ `Library Palette` replaces the Behavior library.

Shockwave audio options

`Xtras` ➤ `Shockwave for Audio Settings` determines the compression setting for Shockwave audio. `Xtras` ➤ `Convert WAV to SWA` (Windows only) creates SWA files from WAVE files. On PowerMacs, SWA files are created using SoundEdit 2.0.7 or Peak LE.

Third-party Xtras

Some third-party Xtras such as Beatnik Lite (in D7), PrintOMatic Lite, and ScriptOMatic Lite (in D6) appear under the `Xtras` menu, providing access to custom help files, *About* boxes, and registration information. The Lingo Scripting Xtras that constitute these products don't appear in the `Xtras` menu, but are accessible via Lingo.

Tool Xtras and libraries

You can add your own Director movie Tool Xtras and Cast Libraries to the `Xtras` menu by placing them in the *Xtras* folder. Director for both Macintosh and Windows recognizes any file in the *Xtras* folder with the extensions .DIR, .DXR, .DCR, or .CST (but not .CCT or .CXT). Director 7 for Macintosh also recognizes files with the proper File Types (`MV07`, `M!07`, `FGDM`, or `MC07`), regardless of their extensions or names. Director 5 and 6 for Macintosh used the older File Types `MV97`, `M!97`, `M*97`, `MV95`, `M!95`, and `M*95`. In D5 and D6, Behavior Libraries placed anywhere in the *Xtras* folder will show up in the `Xtras` menu. In D7, they should be placed in the *Xtras/Libs* folder and will appear under the `Window` ➤ `Library Palette`.

Do You Need an Xtra?

Ask around before assuming that you need an Xtra. There is often a built-in Lingo command or Xtra that comes with Director that solves the problem. Macromedia technical support or a competent Lingo consultant can tell you whether Director can accomplish the desired task.

Xtras are not a replacement for proper use of Director or Lingo, nor are they an evil to be avoided.

Identify the Xtras you'll need early in a project. If too many requirements cannot be handled by Director, you may be better off with another tool. Whatever the circumstance, your options become limited if you wait until the last minute (as many do). Allow extra time to research, obtain, implement, and test Xtras.

Don't underestimate the value of managing your client's (or your own) expectations. Many clients would be better served if problematic features were simply dropped or handled via an installer, *Read Me* file, documentation, or training.

Reasons to use Lingo instead of Xtras:

- Lingo is free with Director. Many Xtras cost money, although many are also free or very inexpensive. Some have licensing distribution fees.
- Lingo-only solutions are much more likely to work across all platforms and in future versions of Director and Shockwave. Xtras may need to be rewritten for future operating systems or platforms.
- Shockwave requires Xtras to be downloaded and installed separately (the SW6 download inclues the Flash Asset and SWA Xtras, and the NetLingo, Sound Import Export, PICT, BMP, GIF and JPEG Xtras' capabilities are built into the Shockwave plug-in).
- Shockwave 7 allows automatic downloading of Xtras, but Lingo-only solutions will support Windows 3.1 and 68K Mac users whom SW7 does not support. The Shockwave 7 download includes the Flash, Font, Text, NetLingo, Multiuser, and SWA Xtras, plus built-in support for sounds, GIF, PICT, BMP, and JPEG assets.
- Lingo and Director tend to be more thoroughly tested, more compatible, and more stable than most Xtras. It is sometimes hard to find or obtain an Xtra on short notice, and unwise to introduce one unless sufficient time is available for testing.

Legitimate reasons to use Xtras:

- You're sure that Lingo alone can't do what you want. Sometimes an Xtra drastically simplifies a task that is truly painful via Lingo alone.
- Director is too slow to perform a mission-critical task. Lingo can create databases and manipulate text, but it is not optimized for either.
- You need a new transition type or sprite asset type not supported by Director.

Poor reasons to use Xtras:

- You are having performance or memory problems. An Xtra isn't going to speed up asset loading or reduce memory usage in most cases.
- You don't know how to accomplish something in Director or Lingo. C programmers and inexperienced Linguists often want to use Xtras where Director and Lingo handle the task adequately, even admirably.
- You are stretching Director beyond any logical boundaries. Director is not well-suited for some tasks, no matter how many Xtras you pile on top of it.

Obtaining Xtras

Xtras are available from various sources:

Macromedia
: Macromedia includes numerous Xtras with Director, including the ActiveX, PowerPoint, Custom Cursor, Flash, and Java Xtras sold separately prior to D6.5.

Third-party Xtras
: There are a wide variety of third-party Xtras available from dozens of companies. Many are sold commercially or as shareware, but some are freeware or donationware.

Custom Xtra development
: You may contract with someone to develop an Xtra or develop one in-house. Some Xtra developers will customize their existing commercial Xtras for a separate fee.

If using an Xtra for a make-or-break portion of your product, allow plenty of extra time and money. Custom Xtra development can be expensive and fraught with delays.

Resources for finding Xtras

There is no single resource that lists all available Xtras, and the list is constantly growing. Try the usual search engines using keywords such as *Macromedia*, *Director*, and *Xtras*, in addition to using the following resources:

The Director 7 CD
: The Director CD includes samples and demonstrations of many third-party Xtras in the *Goodies* and *Xtra Partners* folders. These include Xtras to play MPEG video, manage large databases, and so on. See also previous Director CDs and the Xtras CD distributed at the 1997 Macromedia User Conference. The Director 6 CD includes unsupported Xtras and sample files for Xtra developers. See the *Xtras\Win* and *Goodies\Director\SoundXtr\Xtras* subfolders under *Macromedia\XDK_d6a4* on the D6 CD.

Books
: Many third-party Director books include CDs with demos of various Xtras.

The Director Xtras Book by Rich Shupe (Ventana/Coriolis)
: Rich does a nice job of covering a wide variety of Xtras, and this book is a great place to start your research without trawling the Web. Rich's site (*http://www.fmaonline.com)* has many links to Xtra developer's sites and the book includes a CD-ROM with many demo Xtras.

Xtravaganza! by Chuck Henderson (Macromedia Press/PeachPit)
: Covers Xtras for Director and other Macromedia products. Includes descriptions of 460 Xtras and a CD-ROM with over 100 demos.

Macromedia's web site is a good (but not comprehensive) place to look for third-party Xtras:

http://www.macromedia.com/software/xtras/director

Zeus Productions offers Xtras that open external documents and launch external applications, and custom Xtra development:

http://www.zeusprod.com

UpdateStage sells Xtras from Red Eye Software (Scott Kildall) and Dirigo Multimedia (Glenn Picher), formerly sold via g/matter, plus many other Xtras:

http://www.updatestage.com/xtras

Media Lab sells some of the consistently most useful and coolest Xtras, including Photocaster, Alphamania, and Effector Set:

http://www.medialab.com

Penworks sells a number of utility Xtras, including CastEffects and Iconizer:

http://www.penworks.com

DonationWare (inexpensive utility Xtras for a small donation) can be found at:

http://www.trevimedia.com/donationware.html

Kent Kersten's Little Planet freeware utilities including FileXtra and ScrnXtra are located at:

http://www.littleplanet.com/kent/kent.html

TreviMedia runs an Xtra-related email list (Xtras-L) where you are free to discuss Xtras from any vendor. Send the following in the body of an email to *listserv@trevimedia.com*:

```
SUB XTRAS-L yourFirstName yourLastName
```

Database Xtras

The following Xtras purport to provide database or text access in some form. I have no firsthand knowledge of these Xtras, but have heard frequent praise for V12 and DataGrip. As with all Xtras, your mileage may vary, so ask around.

V12 Database Engine from Integration New Media (cross-platform):

http://www.integration.qc.ca/

DataGrip (MS Access databases, Windows only):

http://www.datagrip.com/

FileFlex (cross-platform):

http://www.fileflex.com

Active XtraBase from Prime Arithmetics (Windows NT servers):

http://www.primearithmetics.com

OpenDBC from Brummell Associates (Windows only):

http://www.btinternet.com/~brummell/

MHTsearch (indexing documents and searching for words):

http://www.meetinghousetech.com

TextCruncher by Yair Sageev (cross-platform text parsing and manipulation):

http://www.itp.tsoa.nyu.edu/~student/yair/texcruncher/HTML/YairTextCruncher.html

Printing Xtras

PrintOMatic (cross-platform, requires scripting):

http://www.printomatic.com

mPrint (Windows only, visual page layout, no scripting):

http://www.mediashoppe.com

zPrint (Windows only, prints external files with external applications):

http://www.zeusprod.com/products/zopen.html

Utility Xtras

The following Xtras have many OS-level functions for each platform.

Buddy API (cross-platform):

http://www.mods.com.au/budapi

DirectOS Xtra (Windows only):

http://www.directxtras.com/do_doc.htm

OSutil Xtra (Macintosh version is shipping, Windows version is imminent):

http://www.magna.com.au/~farryp/director/xtras/

Shopping for Xtras

The purchase price of an Xtra can be dwarfed by the cost of the time spent trying to implement it or the time lost if it doesn't work. When you purchase an Xtra, ask the vendor to recommend someone to help you implement it if necessary. Assuming you *need* an Xtra, consider the following criteria:

Do you need an Xtra and does it do what you need?
: You must identify the problem before you can decide whether an Xtra will solve it. Obtain a demo version of the Xtra and test whether it solves your problem. Director 7 supports many new features previously requiring a third-party Xtra.

Does the Xtra support the desired platforms?
: It is not unusual for an Xtra *not* to support all the platforms you intend to support. Even so, you may not need an Xtra on all platforms or may use Xtras from different vendors on different platforms. Beware of Xtras that run so poorly

as to be unusable on older platforms. It is increasingly common that Windows Xtras don't support Windows 3.1. Many do not yet fully support Windows NT. Some Xtras don't support older 68K Macintoshes.

A so-called *cross-platform* Xtra requires separate versions for each platform. Only Xtras written in pure Lingo can use the same file on multiple platforms. Verify that Xtras are compatible with D7.

Is it an Xtra or XObject?

Although D6 supports some older XObjects, XObjects will not work under Windows NT when using a 32-bit Windows Projector, nor are XObjects supported in D7.

Does the Xtra work the same on all platforms?

Many Xtras, such as Macromedia's FileIO Xtra, use identical *commands* on both platforms, but may not *operate* identically in all respects. Anticipate variations across platforms, especially when dealing with external files, applications, or hardware.

Does the Xtra support the desired version(s) of Director?

Many Xtras work in multiple versions of Director, but some do not. Many Xtras that worked in D6 will not work in D7 due to major architectural changes. You may need an updated version of existing Xtras.

Is the Xtra "Shockwave-safe"?

Shockwave 7 recognizes only Xtras marked by the developer as "Shockwave-safe." Any Xtra that provides file access or system API access would be a security risk. Even innocent Xtras that worked with SW6 must be recompiled to work with SW7.

Is the price justified by the added capability?

Xtra prices range from the free to the exorbitant and there is not necessarily a correlation between price and quality. Free Xtras are rarely supported and are less likely to be updated in the future, and even expensive Xtras may be justified by the time they'll save you. Most Director users feel that Xtras are expensive, but a working Xtra is almost always cheaper than custom development. If the Xtra developer provides good support, you are buying a solution, not just an Xtra.

What expertise is required?

Most Lingo scripting Xtras require an average Linguist. Sprite Xtras may also require Lingo to control the sprite asset, but many Xtras require little or no scripting.

Are there good examples and documentation?

Even the simplest Xtra may be hard to use without an example, and even a complicated Xtra can be easy to implement with a proper example. Most companies provide electronic documentation ranging from the inadequate to the excellent. Check their web site and inquire about documentation before you buy. Third-party documentation is sometimes available (see the Xtras books cited earlier).

Always try to obtain a demo version for testing, but do *not* wait until the last minute to purchase and test the live version. Sometimes an Xtra drastically simplifies a task that is truly painful via Lingo alone.

Who is the developer or publisher and who provides support?

Third-party Xtras are sold both directly by developers and via separate publishers and distributors. Some vendors and their products have earned excellent reputations while others have earned poor ones. Ask around.

Support also ranges from non-existent to excellent, and may be provided via phone or exclusively via email. Ask about refund policies, and ask who provides the technical support (engineers, salespeople, the developer, the distributor, or the publisher). Reputable vendors will offer a money-back guarantee.

Can you obtain the Xtra easily and in a timely fashion?

You can't use an Xtra if you need to ship tomorrow and it takes a week to obtain a password. Most Xtras are available electronically but are *not* available in packaged versions or through typical retail or catalog channels. Ask vendors if they provide free downloads of demo versions, accept payment forms convenient to you, and process orders promptly.

What are the licensing requirements?

Most Xtras are designed to be shipped with your Projector (although there may be different versions for authoring and runtime use). Some Xtras require a royalty for each copy distributed, but most charge either a flat rate for an unlimited number of copies, or on a per-product basis.

How long has the Xtra been available?

If an Xtra has been around for some time, it may be more robust or better documented, or it may be painfully obsolete. Whenever a new version of Director or an OS comes out, find out whether the Xtra has been tested or upgraded. You can reasonably expect Macromedia Xtras to work with the current version of Director and latest OS version, although, for example, the Custom Button Xtra is obsolete in D7.

XTRAINFO.TXT

The *XTRAINFO.TXT* file has several purposes. In D6, it determines which Xtras are included when using the *Check Movie for Xtras* and *Include Network Xtras* checkboxes under `File➤Create Projector➤` *Options*. In D7, it determines the Xtras added by the *Add Defaults* and *Add Network* buttons under `Modify➤Movie➤Xtras` and whether Xtras are downloadable. It also translates the names of Xtras listed under `Modify➤Movie➤Xtras` across various platforms.

XTRAINFO.TXT should be placed in the same folder as the Director application (not in the *Xtras* folder). It is used when opening files cross-platform and when a Projector is created. It need not be shipped with the Projector.

Format of XTRAINFO.TXT

Each entry in *XTRAINFO.TXT* specifies a property list defining the name of the Xtra file to be used for each platform, such as:

```
[#name:"XtraName" {, #name:"XtraName",...}, #type:#xtraType]
```

where *#name:"XtraName"* is one of the following symbols followed by the name of the Xtra used for the particular platform (only *#namePPC* and *#nameW32* are supported in D7):

```
#name68K: "Mac 68K Xtra"
#namePPC: "PowerMac Xtra"
#nameFAT: "FAT Mac Xtra"
#nameW16: "Win16.X16"
#nameW32: "Win32.X32"
```

The entry for the Windows 16-bit version of the FileIO Xtra in D6.0 was misspelled as *#namwW16* and should be *#nameW16*. See the comments in *XTRAINFO.TXT* (which differs markedly in D6 and D7) for additional information.

Using XTRAINFO.TXT in D6

In D6, *#type:#xtraType* in *XTRAINFO.TXT* specifies the type of the Xtra:

#type:#asset
: Sprite or Transition Xtra

#type:#lingo
: Lingo Scripting Xtra

#type:#mixin
: MIX Xtra, included if *Check Movies for Xtras* is checked

#type:#mix
: Other MIX Xtra, included only if used by linked asset

#type:#net
: Network Xtra, included if *Include Network Xtras* is checked

#type:#netlib
: WinSock Library for PPC only

#type:#service
: MIX Services, included if *Check Movies for Xtras* is checked

See the comments in *XTRAINFO.TXT* (which differs markedly between D6 and D7) for additional information

In the `File` ➤ `Create Projector` ➤ *Options* dialog box are two checkboxes:

Include Network Xtras
: Includes all Xtras marked as *#type:#net* in *XTRAINFO.TXT*, but does not include those marked as *#type:#netlib*. See Table 10-2.

Check Movie for Xtras
: Includes all Xtras marked *#type:#mixin* or *#type:#service* in *XTRAINFO.TXT*, but does not include those marked as *#type:#mix*. (See Table 10-3.) This option also includes any Xtras listed under `Modify` ➤ `Movie` ➤ `Xtras` from any of the movies being bundled into the Projector.

Deselecting both checkboxes creates a Projector that contains only the Director movies, castLibs, and Xtras added manually via the file picker in the *Create Projector* dialog box. Xtras can still be distributed separately in an *Xtras* folder to be included with the Projector.

The types specified in *XTRAINFO.TXT* can be edited to your liking. Notice that the GIF Import and JPEG Import Xtras are listed as both *#net* and *#mix* Xtras.

To include an arbitrary Xtra in your Projector when the appropriate checkbox is checked, simply add it to the *XTRAINFO.TXT* file with a *#type* of *#net*, *#mixin*, or *#service*.

Using XTRAINFO.TXT in D7

The entries in the D7 *XTRAINFO.TXT* file include the following optional properties, which supersede the *#type* property specified in D6:

#type: #default
: Included by default with every new movie

#net: #xtra
: Included by *Add Network* button

#net: #netLib
: Not included automatically

#info: "url"
: URL for more information about downloadable Xtras

#package: "url"
: URL from which Xtra package can be downloaded

These attributes affect the following options under `Modify ➤ Movie ➤ Xtras` (see Figure 10-1):

Add Defaults
: Adds Xtras flagged as *#type: #default* to *the movieXtraList*.

Add Network
: Adds Xtras flagged as *#net: #xtra* to *the movieXtraList*.

Info
: Displays information about the Xtras from the URL specified by the *#info* property.

Include in Projector
: Allows Xtras to be included or excluded individually (unlike in D6) when this movie is added to a Projector. (I recommend against this.)

Download if Needed
: Allows an Xtra to be downloaded from the URL specified by the *#package* property. Macromedia packages downloadable Shockwave-safe Xtras beyond those included in the standard Shockwave installation (most notably the QT3 and Animated GIF Asset Xtras, and XML parser). Many third-party developers also make Xtra packages available for download.

Standard Macromedia Xtras

The following sections list the Xtras that come standard with Director. Note that some are not available for all platforms.

Network Xtras

Table 10-2 lists the Xtras that provide network services, such as http and ftp access, and many net-related Lingo commands. The Xtras of type *#net* (but not *#netlib*) are included in the Projector automatically if the *Include Network Xtras* option is checked when creating a D6 Projector. D7 includes all the Xtras in Table 10-2, except WinSockLib, with Projectors by default. You must include the *NetManage WinSock Lib* manually* when shipping a PowerPC or Fat Mac Projector that accesses the Internet.

Table 10-2: Network Xtras Filenames

Type[1]	**PowerPC**	**Mac 68K**	**Win 32**	**Win 16**
#net	GIF Import	GIF Import 68K	GIF Import.X32	MixGIFf.X16
#net	JPEG Import	JPEG Import 68K	JPEG Import.X32	MixJPEG.X16
#net	INetUrl PPC Xtra	None[2]	INetURL.X32	INetURL.X16
#net	NetFile PPC Xtra	NetFile 68K Xtra	NetFile.X32	NetFile.X16
#net	NetLingo PPC Xtra	NetLingo 68K Xtra	NetLingo.X32	NetLingo.X16
#netlib	NetManage WinSock Lib (D6); WinSockLib (D7)	None[2]	None[3]	None[3]

[1] The GIF and JPEG Import Xtras are listed twice (as both *#net* and *#mix* Xtras) in D6's *XTRAINFO.TXT* file.
[2] The code for the *INetUrl* Xtra and *NetManage WinSock Lib* for 68K Macs is built into Standard and FAT Macintosh Projectors, and does not require an Xtra on 68K Macs or when using Standard (non-native) Projectors on PowerPCs.
[3] WinSock support under Windows is provided by DLLs installed with the OS or by the user's browser in the Windows *System* directory.

Use the Lingo property *the netPresent*, not the *netPresent()* function, to check for the presence of the NetLingo and NetFile Xtras. Shockwave movies do not require these Xtras.

MIX import Xtras

Table 10-3 lists the MIX Xtras that allow Director to import and export various graphic file formats (GIFs, PICTs, and so on). Xtras shown for both PowerPC and Mac 68K are Fat Binary Xtras. The Xtras of type *#mixin* and *#service* (but not *#mix*) are automatically included in the Projector if the *Check Movies for Xtras*

* John Taylor reports that when using D6, the Power Macintosh NetManage WinSock Lib can conflict with a file of the same name installed by Quicken 98 in the Macintosh *System* folder. He installs the PowerMac NetManage WinSock Lib in both his *Xtras* folder and the Macintosh *System* folder (over Quicken 98's version) to alleviate the conflict. An alternative is to ship a D6 68K Macintosh Projector, which has its own internal WinSock management.

option is checked when creating a D6 Projector. Those of type *#default* are included with D7 Projectors, unless manually removed from the list of Xtras under `Modify ➤ Movie ➤ Xtras`.

Table 10-3: Graphic MIX Xtras by Platform

D6 Type[1]	D7 Type	PowerPC	Mac 68K	Win 32	Win 16
#service	#default	Mix Services	Mix Services	MixServices.X32 (D7) mix32.X32 (D6)	Mix16.X16
#mix[2]	N/A	BMP Import Export	BMP Import Export	BMP Import Export.X32	MixBMP.X16
#mix[2]	N/A	GIF Export	None	GIF Export.X32	None
#mix	#default	GIF Import	GIF Import 68k	GIF Import.X32	MixGIF.X16
#mix[3]	N/A	None	None	ImageMark Import.X32	None
#mix[2]	N/A	JPEG Export	None	JPEG Export.X32	None
#mix[2]	#default	JPEG Import	JPEG Import 68k	JPEG Import.X32	MixJPEG.X16
#mix	N/A	LRG Import Export	LRG Import Export	LRG Import Export.X32	MixLRG.X16
#mix	N/A	MacPaint Import	MacPaint Import	MacPaint Import.X32	MixMcPnt.X16
#mix	N/A	Palette Import	Palette Import	Palette Import.X32	MixPal.X16
#mix	N/A	Photoshop 3.0 Import	Photoshop 3.0 Import	Photoshop 3.0 Import.X32	MixPS30.X16
N/A	N/A	Photoshop™ Filters	None	Pshopflt.X32	None
#mix[4]	N/A	None	None	Photoshop Clut Import.X32[4]	None
#mix[2,5]	N/A	PICT Import Export	PICT Import Export	None	None
#mix	N/A	PNG Import Export	PNG Import Export	PNG Import Export.X32	MixPng.X16
#mix	N/A	Targa Import Export	Targa Import Export	Targa Import Export.X32	MixTARGA.X16
#mix	N/A	TIFF Import Export	TIFF Import Export	TIFF Import Export.X32	MixTIFF.X16

[1] The Xtra's type is specified in *XTRAINFO.TXT* and can be modified as desired. The GIF Import and JPEG Import Xtras are listed twice (as both #net and #mix Xtras) in D6.
[2] The GIF Export, JPEG Export, PICT Import Export, and Sun AU Import Export Xtras are new in D6.5 and are used only with the Save as Java Xtra. The JPEG Agent and BMP Agent replace the JPEG Import and the BMP Import Export Xtras in D7.0.1.
[3] The ImageMark Xtra imports PCD, PCX, and WMF files, and the TIFF preview from EPS files (see Chapter 4). It is for authoring only and is not licensed for redistribution. Obsolete in D7.
[4] The Photoshop Clut Import.X32 Xtra was added in D6.0.2, excluded in D6.5, and added back in D7.
[5] Remove the PICT Import Export Xtra except when using Save as Java.

MIX Xtras are required to drag-and-drop certain graphic and sound files from the desktop into the Cast, use linked graphics or sounds, or use *importFileInto* (which isn't recommended within Projectors). Refer to Chapter 4 for details on importing various file types. The *MIX Services* Xtra is required when using any of the other MIX Xtras, or any linked file types.

The Shockwave 6.0 plug-in supported only linked media types for which MIX Xtras were installed in the Shockwave support folder. Shockwave 6.0.1 and 7 recognize GIF, JPEG, PICT, BMP, AIFF (compressed and uncompressed), and WAVE (uncompressed only) files automatically without any Xtras. Other linked media types, such as SWA, QT3, and Flash, require Sprite Asset Xtras, but not MIX Xtras or the MIX Services Xtra. See *http://www.zeusprod.com/nutshell/mix.html* for the complete story on MIX versus non-MIX Xtras. Note also the many Sprite Xtras are included with the Shockwave 7 download.

The D6.5 PICT Import Export Xtra conflicts with the PICT Agent Xtra when importing bitmaps. Remove it from the D6.5 for Macintosh *Xtras:MIX* folder except when using `File➤Save as Java`.

Secret agent Xtras

The *agent* Xtras in the *MIX Xtras* folder are used when communicating with external editors during authoring. Agents are implemented as separate Xtras when support for importing a file type is already built into Director. (Agents for other file formats are included in the *Import/Export MIX Xtras*.) Agents are for authoring-time only. They are not needed by Projectors or Shockwave and are not included in *XTRAINFO.TXT*.

Agent Xtras include:

- AVI Agent (Windows only)
- PICT Agent
- QuickTime Agent (D6 only)
- xRes Agent
- JPEG Agent and BMP Agent (new in D7.0.1)

Sound and SWA Xtras

The Sound Import Export Xtra is another MIX Xtra needed whenever using external sounds. By convention, the SWA Xtras are kept in the *Media Support* Xtras folder, not the *MIX* Xtras folder. SWA compression is handled by the SWA Export Xtra for SoundEdit or Peak LE on PowerMacs and the Swacmpr.X32 or SWAcnvrt.X32 Xtra under Director for Windows (32-bit only).

You must include all the Network Xtras if using Internet-based SWA files and include the NetFile Xtra even if using local SWA files only. Include the Mix Services Xtra and Sound Import Export Xtra for any external sounds, including SWA. See Macromedia TechNote #12598 if your Sound Import Export Xtra is not being recognized.

Only the SWA decompression and streaming Xtras shown in Table 15-13 are needed at runtime. The following SWA Xtras are used during authoring only.

Windows 95/98/NT development-time-only SWA Xtras:

- Swasttng.X32
- Swaopt.X32
- Swacnvrt.X32
- Swacmpr.X32

PowerMac Director development-time-only SWA Xtras:

- SWA Compression Xtra
- SWA Options Xtra
- SWA Settings (Dir) Xtra

The SunAU Import Export Xtra is another MIX Xtra used to import .au audio files. See Table 15-13 for the sound-related Xtras needed at runtime.

These SWA Xtras are used by SoundEdit and are kept in the *System Folder:Macromedia:Xtras* folder.

PowerMac SoundEdit Development-Time-only SWA Xtras:

- SWA Export Xtra
- SWA Settings (SE16) Xtra

Macromedia Sprite Asset and Lingo Xtras

Table 10-4 lists additional Sprite Asset and Lingo Xtras from Macromedia that ship with Director 6 and 7 and can be distributed with your Projector. Note that the Cursor, Flash, QuickTime 3, and ActiveX Xtras are new to D6.5. Only the distributable version of each of these Xtras is shown in Table 10-4. The authoring-only versions cited in the footnotes to Tables 10-4 and 10-5 must not be distributed.

Table 10-5 lists the new Xtras in D7 that add new asset types and other capabilities.

Table 10-4: Sprite Asset and Lingo Xtras

D6 Type	D7 Type	PowerPC	Mac 68K	Win 32	Win 16
#asset	Obsolete	Button Editor	Button Editor	Buttoned.X32	Buttoned.X16
#asset	Obsolete	QuickDraw 3D Xtra	None	QD3DXtra.X32	None
#asset[1]	N/A	CursorsPPC (D6) Cursor Asset (D7)	None	Cursor.X32	None
#asset[2]	#default	Flash Asset PPC	None	Flash Asset.X32	None
#asset[3]	#default	QuickTime Asset PPC (D6) QuickTime Asset (D7)	None	QuickTime Asset.X32 (D6); QT3Asset.X32 (D7)	None
#asset[4]	N/A	N/A	N/A	ActiveX.X32	None
#lingo[5]	N/A	FileIOXtraFat (D6) FileIOXtraPPC (D7)	FileIOXtraFat	FILEIO.X32	FILEIO16.X16
#lingo	Obsolete	QTVRXtra	QTVRXtra	QTVRW32.X32	QTVRW.X16
#lingo	Obsolete	N/A	N/A	DirMMX.X32	None

[1] The CrOptPPC and CurOpt.X32 Xtras and *Cursor Behavior Library.cst* are for authoring use only and should not be distributed.
[2] The Flash Asset Options PPC and Flash Asset Option.X32 Xtras and *Flash Behavior Library.cst* are for authoring use only and should not be distributed.
[3] The QuickTime Asset Options PPC and QuickTime Asset Option.X32 Xtras and *QT3 Behavior Library.cst* are for authoring use only and should not be distributed.
[4] The ActxPriv.X32 Xtra is for authoring use only and should not be distributed. The *Wintdist.exe*, *Aprxdist.exe*, and *Axdist.exe* installers can be distributed.
[5] Refer to Chapter 14, *External Files*, in Lingo in a Nutshell for details on the FileIO Xtra.

Table 10-5: New Xtras in Director 7

Usage	Type	PowerPC	Win32
Animated GIF sprite	N/A	Animated GIF Asset	Animated GIF Asset.X32
Animated GIF options[1]	N/A	Animated GIF Options	Animated GIF Options.X32
Vector shapes[1,2]	N/A	Shape Xtra	ShapeXtra.X32
Text creation[1]	N/A	Text Asset Options	TextAuth.X32
Text sprites	#default	Text Asset PPC	TextAsset.X32
Text rendering	#default	TextXtra PPC	TextXtra.X32
XML parsing	N/A	XMLParser PPC Xtra	XMLParser.X32
DXR compression[1]	N/A	LZ77 Compression PPC Xtra	LZCompr.X32
DCR compression[1]	N/A	Squish Rules PPC Xtra	Squish.X32
Multiuser server	N/A	Mulitusr	Multiusr.X32

Table 10-5: New Xtras in Director 7

Usage	Type	PowerPC	Win32
Sound mixer	#default	N/A	MacroMix.X32
Font rendering	#default	Font Xtra PPC	Font Xtra.X32
Font management	#default	Font Asset PPC	Font Asset.X32
Font options[1]	N/A	Font Asset Dialog	Font Asset Dialog.X32
MP3 audio	N/A	MPEG 3 Import Export	MPEG3 ImportExport.X32
QT3 export[1]	N/A	QTExportXtra	QTExport.X32
QT3 sprites	#default	QuickTime Asset	QT3Asset.X32[3]
QT3 editing[1]	N/A	QuickTime Asset Options	QTAuth.X32
DirectSoundMixer[4]	#default	N/A	DirectSound.X32

[1] Authoring only.
[2] Vector shapes require the Flash Asset Xtra for playback.
[3] QT3Asset.X32 includes the QT3Mix Sound Mixer.
[4] New in D7.0.1 and included with SW7.0.1.

Miscellaneous Macromedia Xtras

- PowerPoint Import (new in D6.5). The Import Xtra for PowerPoint converts PowerPoint presentation files into Director movie files.
- Java Export Xtra (new in D6.5). The `File ➤ Save as Java` feature exports Shockwave movies as Java code so they can be played back in browsers without Shockwave. It uses the CompileJavaPPC, JavaConvert, UIHelper PPC Xtra, FileXtra, Sun AU Import Export, PICT Import Export, JPEG Export, and GIF Export Xtras on the Macintosh, and the Javacvnt.X32, UiHelper.X32, Filextra.X32, Sun AU Import Export.X32, JPEG Export.X32, and GIF Export.X32 Xtras under Windows. It uses the *Behavior Library for Java.cst* in D6.5 and the `Window ➤ Library Palette` in D7, and *Save as Java.dxr* files on both platforms.
- The MUI Dialog Xtra creates custom alert and non-modal dialog boxes. See Chapter 15, *The MUI Dialog Xtra*, in *Lingo in a Nutshell*, and *http://www.zeusprod.com/nutshell/chapters/mui.html*.
- The Actor Control and Cast Control Xtras are using during authoring to manage sprite, cast members, and castLibs and are not used in Projectors.
- The Cue Card Xtra prompts the developer to run a tutorial the first time Director 6 is launched.
- D7 for Windows includes new Intel Dynamic filters *WDEadd.X32*, *WDEd2Opt.X32*, *WDEdis.X32*, and *WPEopt.X32*. They appear under the `Insert` menu.
- The D6.5 update CD's *Goodies:Import Goodies* folder includes the unsupported SWA and MPEG3 Xtras, which became official in D7.

- There are numerous unsupported Xtras on the Director 6 CD. Most are simply demonstrations for Xtra developers. See the *Xtras\Win* and *Goodies\Director\SoundXtr\Xtras* subfolders under *X:\Macromedia\XDK_d6a4* on the D6 CD. The CommPort XObject (the Windows analog of the Macintosh Serial Port XObject) is included in the discontinued folder.
- *Xobglu32.DLL* and *Xobglu16.DLL* are a pair of "thunking" DLLs that allow D5 and D6 32-bit Projectors to use older 16-bit XObjects under Windows 95/98. (16-bit XObjects are not supported under Windows NT.) These DLLs are automatically built into 32-bit Windows Projectors and are extracted and used automatically, if necessary. The temporarily extracted files are deleted when the Projector terminates. Search the Macromedia TechNotes for the word "thunk" for details.
- *DLLGLUE.DLL* is an obsolete Windows 3.1 XObject (written by Paul Hamilton) that allowed Lingo to access Windows API functions or DLLs not specifically designed as XObjects. It has been superseded by Xtras such as DirectOS and Buddy API, which access the Windows API, and by RavWare's (*http://www.ravware.com*) GLU32 Xtra for Windows 95/98/NT, which allows Lingo to call the Win32 API.
- The SerialPort and XCMDglue XObjects embedded in Director for Macintosh's resource fork provide serial port access and allow Director to use Hypercard XCMDs in D6, but are obsolete D7.
- The rumored spell-checker and encryption Xtras from Macromedia have yet to materialize.

Third-party Xtras

The third-party Photocaster Lite (*http://www.medialab.com*) and Beatnik Lite Xtra (*http://www.headspace.com*) are included with D7. The third-party PrintOMatic Lite, ScriptOMatic Lite, and PopMenu Lite Xtras are installed by default with D6. The full version of PrintOMatic is now published exclusively by Electronic Ink (*http://www.printomatic.com*) and the PopMenu Xtra has been renamed the Popup Xtra and is available from UpdateStage (*http://www.updatestage.com/xtras*). The full version of ScriptOMatic may be available from *http://www.trevimedia.com*.

There are samples and demonstrations of many other third-party Xtras on the Director 6.0 and 7.0 CDs as described earlier under "Obtaining Xtras."

Loading and Registering Xtras

The preferred way to load Xtras is to include them in the appropriate *Xtras* folder where they will be opened and closed automatically by Director.

You should not access Xtras manually using *openXlib* and *closeXlib*. Use *openXlib* and *closeXlib* with XObjects only.

You can include Xtras in the resource fork of Director for Macintosh or a Macintosh Projector. This is not recommended for Xtras, but was commonly done with the FileIO XObject prior to Director 5.

Director 6 and 7 support bundling Xtras into Projectors on both Macintosh and Windows. I recommend against bundled Xtras, though.

When an Xtra is loaded, Director registers the Xtra in an internal dictionary and identifies it by its unique Xtra ID (and optional Xtra version number) assigned by the Xtra developer.

Director decides whether to load Xtras based on their external file names (under Windows) and File Types (on the Macintosh), but identifies Xtras by their internal Xtra IDs that are independent of the external filename or type.

Director will issue an error message if it encounters two Xtras with the same Xtra ID, unless one has a later version number. In practice, many Xtra developers omit version numbers. It is best to delete old versions of Xtras and include only the current version in your *Xtras* folder. In some cases, a developer may inadvertently forget to generate a unique Xtra ID when copying a template from one of Macromedia's Xtras. Remove suspicious Xtra(s) from all *Xtras* subfolders (and restart Director) until you locate the culprit. (For Director 5, be sure to check in the multiple Xtras folders described under "The Xtras Folders" later in this chapter.)

When Director requires a particular Xtra, it consults the dictionary to determine whether the Xtra has been registered. When Director loads a movie, it determines whether any cast members require a custom Sprite Xtra. If so, and the Xtra is not present, Director displays an error message.

If a required Sprite Xtra is missing, Director will display a red X on the Stage in place of the sprite. If a Transition Xtra is missing, Director will perform a jump-cut instead of the transition. If a MIX Xtra is missing, Director will not be able to import or use linked assets of the particular type (linked sounds won't play and linked images won't appear).

If a Lingo Xtra is omitted, Director will post a "Handler not Found" error because the required command won't be recognized. For example, if you omit the NetLingo Xtra, net-related Lingo commands such as *downloadNetThing()* will fail.

Once an Xtra is loaded, it is available for the duration of the Projector, not just the current movie. The methods within a Lingo Scripting Xtra can be accessed from any other script.

There are several reasons that an Xtra might not load or register:

Xtras are read only at startup
: You need to restart Director or the Projector to load new Xtras.

Corrupted Xtras cache file
: When in doubt, delete the Xtras cache file and try again.

Corrupted Xtra file
: A corrupted Xtra may be caused by a download problem, a disk error, or an incomplete update of Director. Replace any suspicious Xtras with a fresh copy.

Xtras weren't automatically bundled

I recommend against bundling Xtras, but if you choose to do this, it occurs automatically only if the movie is included in the Projector or if the appropriate options are checked under `File ➤ Create Projector ➤ Options` in D6. In D7, the *Include in Projector* option must be checked for the relevant Xtra under `Modify ➤ Movie ➤ Xtras`.

Wrong version of Xtra

Obtain and use the correct Xtras for D7. For example, the D6.5 QT3 Asset Xtra won't work in D7.

Xtra is not Shockwave-safe

Shockwave 7 will only recognize Xtras compiled as "Shockwave-safe" by the Xtra developer.

Not enough memory

For example, the QuickTime Asset Xtra for Macintosh requires that about 15 MB RAM be allocated to Director. The default memory allocations may suffice, but you should probably increase the default allocation when using QuickTime 3 with Macintosh Projectors.

Xtra requires another Xtra that is missing

The Sound Import Export Xtra and other MIX-related Xtras require the MIX Services Xtra. The error may say that the Sound Import Export Xtra is not loaded, but you actually need to add the MIX Services Xtra to your *Xtras* folder.

Sound Import Export Xtra won't load

This is a known issue addressed by Macromedia TechNote #12598. Also, under Windows NT, rename your Xtras to obey the 8.3 naming convention to ensure that they load properly. The internal name of the Xtra is unaffected.

Xtra requires a missing system component

For example, the Macintosh QuickTime Asset Xtra won't load unless version 3.x of the Macintosh QuickTime extension is installed. And the QTVR Xtra requires a full QT installation, including the QTVR components.

Xtra requires a missing DLL

For example, the Buddy API Xtra requires the accompanying *BUDAPI32.DLL* or *BUDAPI16.DLL* in the same folder. Other Xtras depend on Windows system DLLs, which may in turn depend on other DLLs. If the Xtra registers on some machines but not others, it is a likely culprit. See the next section, "Determining which DLLs are needed."

Conflicting class ID (known as a GUID)

If two Xtras have the same internal class ID, the first one encountered will load, but the second will generate a "Duplicate Xtra" error message and will not load. If you are an Xtras developer, generate a unique ID for your Xtra as described in the MOA documentation. There is no foolproof way to tell which Xtra is the problem, but you can sort all Xtras by name, then make an educated guess. Remove Xtras one by one until you find the two culprits, which may have different names or reside in different subfolders. You won't get the "Duplicate Xtra" error when at least one of them has been removed from the *Xtras* folder.

Wrong filename or File Type

Macintosh Xtras must have the correct File Type (`Xtra`) and Windows Xtras must have an .X16 or .X32 extension. XObjects are not opened automatically (unless embedded in the Macintosh resource fork). XObjects must be opened with *openXlib* instead.

Wrong Xtra for Projector type

Windows 3.1 Projectors require 16-bit Xtras and Windows 95/98/NT Projectors require 32-bit Xtras require 32-bit Projectors. 16-bit Windows Xtras cannot be tested from the D6 authoring environment. You need to test with a D6 Projector or the D5 authoring environment.

Wrong processor or insufficient FPU

Many Xtras won't work on 68K Macintoshes and won't load unless running on a PowerPC. The SWA Xtras require a PowerPC during authoring and at least a 68040 with an FPU (floating-point unit) for playback.

Wrong folder location or folder nested too deep

Projectors don't load Xtras from the same folder as the authoring environment and Shockwave can only access Xtras in the Shockwave plug-ins folder. See "The Xtras Folders" later in this chapter for details on where Xtras must be located. Xtras nested more than five folders deep within the *Xtras* folder will not be found.

Xtras for authoring only

Many Xtras, as a security measure, will refuse to register when used with a Projector. Some require that a registration method be called from Lingo before they can be used in a Projector. Tool Xtras do not register in a Projector. Macromedia's Sprite Xtras come in two different flavors, one for authoring and one for Projector (runtime). See Tables 10-4 and 10-5.

Bundled Xtras interfere with Xtras folder

There have been rumors that bundled Xtras prevent Director from loading Xtras from the *Xtras* folder. I recommend against bundling Xtras.

The proper DEF entries are omitted

If you are an Xtras developer, the Xtra will not be recognized unless you export the proper DEF entries (and remember to compile your DEF). Use `dumpbin /exports` (requires MSVC Developer's Studio) to show what your Xtra exports. If the exports are wrong, the Xtra won't register on any machine.

Determining which DLLs are needed

An Xtra's developer should tell you which DLLs, if any, are required for the Xtra. If not, use `dumpbin` (included with MSVC Developer's Studio) with the `/imports` option to identify the needed DLLs.

Here is the output for Buddy API, filtered to highlight the needed DLLs:

```
G:\MSDEV\bin\dumpbin /imports G:\BudApi\budapi.X32 | find "dll"
      KERNEL32.dll
      USER32.dll
      ADVAPI32.dll
      budapi32.dll
```

You can repeat the process to determine the DLLs that *budapi32.dll* requires "downstream":

```
G:\MSDEV\bin\dumpbin /imports G:\BudApi\budapi32.dll | find "dll"
      kernel32.dll
      user32.dll
      oleaut32.dll
```

Note that future versions of Buddy API reportedly will not require a separate DLL.

The Xtras Cache and Director Preferences

In authoring mode, Xtras are cached, so that Director doesn't need to reregister them every time it starts up (although an Xtra can be designed to allow itself not to be cached). However, all installed Xtras are registered every time a Projector starts (that is, Projectors never cache Xtras). Most Xtras register silently, although demo versions may post a warning dialog box when registering. To perform a clean test of an Xtra, run it from a Projector. If you are having troubles with Xtras, try deleting the Xtras cache file and restarting Director.

On the Macintosh, the cache file is named *dirapi.mch* in D7. In prior versions it is named *Director 6.0 Xtra Cache* (or *Director 5.0 Xtra Cache*) and is located in the Macintosh *System Folder:Preferences* folder. The Preferences file is named *Director 7.0 Preferences*, *Director 6.0 Preferences*, or *Director 5.0 Preferences* in the same folder.

Under Windows, the cache file is named *D70Xtra.MCH*, or *D60Xtra.MCH* and is located in the Director application's folder. (A separate *D50Xtra.MCH* cache file is located in the folders where the 16-bit and 32-bit versions of Director 5 for Windows are installed.) There is no separate D7 preferences file under Windows. Preferences are stored in the Windows Registry file. The D6 and D5 preferences file is named *Director 6.0 Preferences.prf* or *Directr5.prf*, and is in the same folder as Director.

Windows Xtras

There are usually different versions of an Xtra for Windows 3.1 and Windows 95/98/NT, but some Xtras do not support all versions of Windows. The type of Windows Xtra needed depends on the Projector's type, not the version of Windows.

16-bit Projectors require 16-bit Xtras (with an .X16 extension) and 32-bit Projectors require 32-bit Xtras (with an .X32 extension).

Only 32-bit Xtras are used during authoring in Director 6 and 7. In Director 6, you can test 16-bit Xtras only from a Windows 3.1–style Projector (which can be tested under Windows 95).

Windows 3.1

Windows 3.1 requires a 16-bit Projector and 16-bit Xtras. It cannot run 32-bit Projectors. Director 6 and 7 authoring is not supported under Windows 3.1. There is no 16-bit version of authoring-time only Xtras, some of which are shown in Table 10-3. Most of the newer Xtras shown in Table 10-4 are not supported under Windows 3.1. None of the D7-specific Xtras in Table 10-5 is supported under Windows 3.1, as D7 does not support Windows 3.1 at all.

Windows 95/98

A 16-bit projector running under Windows 95/98 requires 16-bit Xtras. A 32-bit projector (the preferred choice) running under Windows 95/98 requires 32-bit Xtras.

Windows NT

32-bit Projectors and 32-bit Xtras are strongly recommended under Windows NT 4.0. Under NT, 16-bit Projectors run under a Windows 3.1 *virtual machine* known as *Windows on Windows*, and use 16-bit Xtras. Windows NT 3.5.1 is not supported as an authoring platform, but will play back 16-bit and 32-bit Projectors with the proper extensions installed. Older 16-bit XObjects are not supported under Windows NT 4.0 when using a 32-bit Projector.

Windows Xtras written in C are only recognized if they have the file extension .X16 or .X32. Both 16-bit and 32-bit Xtras can share the same folder without conflicts. The correct version of the Xtra will automatically be opened to match the Projector (16-bit or 32-bit) regardless of the Windows version in use. Tool Xtras built in Lingo with a .DIR, .DXR, .DCR, or .CST extension are recognized during authoring. Older Windows XObjects used a .DLL extension, but included a message table not found in typical DLLs. Use the GLU32 Xtra to access an arbitrary Windows DLL or API call from Director.

Macintosh Xtras

The type of Xtra needed on the Macintosh depends on the Projector type and not the processor type (68K versus PowerPC). Xtras for both PowerPCs and 68K Macs can reside in the same folder without conflict.

If using a Standard (68K) Macintosh Projector on a PowerPC, both the Projector code and Xtra code run under emulation. Providing both PowerPC and 68K Xtras or a single Fat Binary Xtra is the simplest solution. Some authoring Xtras, such as SWA Compression, are not supported on 68K Macs. See also Table 7-1.

Macintosh Xtras are recognized only if they have the proper case-sensitive four-character File Type (`Xtra`) and Creator Code (`Xown`). If you've downloaded an Xtra and it is not being recognized, set its File Type and Creator Code using ResEdit or similar tool. Tool Xtras built in Lingo with a .DIR, .DXR, .DCR, or .CST extension are recognized during authoring as are files with the corresponding Mac File Types shown in Table 4-4. Older Macintosh XObjects used the File Type `XOBJ` and Creator Code `MMDR` or `MD93` and must be opened manually with *openxlib*. XObjects are not supported in D7. Jason Winshell (*jwins@slip.net*) can convert some older XObjects to Xtras without requiring the source code.

The Xtras Folders

Xtras usually reside in an *Xtras* folder where Director or your Projector will find and open them automatically. The location of the *Xtras* folder depends on the platform, the version of Director, and whether you are running the authoring environment, a Projector, or in Shockwave.

Director for both Macintosh and Windows automatically load Xtras within the *Xtras* folder located below the folder where Director is installed. Projectors automatically load Xtras within the *Xtras* folder located in the folder containing the

Projector. Newly installed Xtras are not recognized until you restart Director or the Projector. (The exception is Xtras downloaded automatically at runtime in D7, which are loaded when the next *goToNetMovie* command is issued.)

You can include both Macintosh and Windows Xtras in the same folder, and Director will load only the appropriate Xtras according to the rules described earlier. Any Xtras inappropriate for the platform or Projector (or any non-Xtras) in the folder are ignored but slow down Director as it checks each file.

Some Xtras install their HTML help files in the *Xtras* folder. Too many files of any type in the *Xtras* folder slows down Director's startup and file saving. Move unused Xtras and any non-Xtra files to other folders to improve performance.

Director and Projectors will scan for Xtras five folders deep below the main folder (don't nest them too deeply, and beware of extremely long folder paths that may exceed the OS limits). Director recognizes *aliases* (Macintosh) and *shortcuts* (Windows) within the *Xtras* folder that point to Xtras elsewhere. See also the TechNote "Installing and Using Xtras" at *http://www.zeusprod.com/technote/xtrainst.html*.

Table 10-6 lists folders that the Director 5 authoring environment checks for Xtras *in addition to* the *Xtras* folder where Director is installed. Director 5 Projectors check only within the *Xtras* folder in the same folder as the Projector, as do Director 6 and 7 Projectors.

Table 10-6: Supplemental (D5) or Temporary Xtras (D6) Folders

Platform	**Xtras Folder**
Windows 3.1	C:\Windows\Macromed\Xtras
Windows 95/98/NT	C:\Program Files\Common Files\Macromedia\Xtras
Macintosh	MacHD:System Folder:Macromedia:Xtras

Macromedia's *Using Director* manual incorrectly claims that Director 6.0 recognizes the secondary *Xtras* folders that are recognized only by Director 5.

In a strange twist, Director 6 and 7 Projectors running from read-only (locked) media will unbundle (i.e., unpack) their Xtras temporarily into the same folders shown in Table 10-6. (I've read reports that they are unbundled into invisible temporary folders, but this does not appear to be true.) Other Macromedia applications, notably SoundEdit, may also use these folders.

D7's new Slim Projectors read Xtras from any *Xtras* folder where the Projector is installed, and also from the Shockwave 7 *System Xtras* folder.

When using Director 5, you can test 16-bit Xtras using the 16-bit authoring environment (even under Windows 95), and 32-bit Xtras in the 32-bit authoring environment. Install the 16-bit Xtras in the *Xtras* folder in which the 16-bit version of Director 5 is installed, and the 32-bit Xtras in the *Xtras* folder in which the 32-bit version of Director 5 is installed.

The Xtras menu

Tool Xtras or Libraries within subfolders within the *Xtras* folder appear in submenus (named for the subfolders) under the `Xtras` menu. Use a minus sign to begin a filename to prevent it from appearing in the `Xtras` menu. In D7, Behavior Libraries should be placed in the *Xtras/Libs* folder, where they can be accessed via `Window ➤ Library Palette`.

MIX, Sprite, Lingo, and Transition Xtras are unaffected by the *Xtras* subfolder in which they are placed, as long as they are in a recognized folder. Table 10-1 shows where each type of Xtra appears in the interface.

The SWA Export Xtra used by SoundEdit and Peak LE doesn't appear in Director's `Xtras` menu.

Xtras for Shockwave

Digitally signed packages can be downloadable dynamically in D7 and SW7. Upon acceptance by the user, the Xtras within the package are installed automatically in the Shockwave 7 *System* folder where they are accessible by Shockwave, ShockMachine, and Slim (system) Projectors. Prior to SW7, the user must download any required Xtras and place them in the browser support folder manually. See the following two sections and Chapter 11 for details.

The Shockwave 6.0 plug-in supports only linked media types for which MIX Xtras are installed in the Shockwave Xtras folder. The Shockwave 6.0.1 plug-in recognizes BMP, PICT, GIF, JPEG, AIFF, AIFC, and WAVE files without requiring any MIX Xtras, but *ignores* any installed MIX Xtras.

The Flash Asset and SWA playback Xtras are not considered MIX Xtras. They are included with the Director for Shockwave download and allow the SW plug-in to play Flash and SWA assets on capable machines.

Only Xtras marked by the Xtra developer as Shockwave-safe are recognized by SW7. Downloaded Xtras are available to all SW movies, not just the one that downloaded them. See *http://www.zeusprod.com/nutshell/xtras.html* for details on Xtra packaging and downloading.

The Shockwave 7 download installs numerous Xtras. Shockwave includes all the functions built into the NetLingo and other net-related Xtras. They need not be included with Shockwave and are necessary only when using a local Projector that accesses Internet-based content. Shockwave 7 supports the same default graphics and sound types as SW6.0.1, but also recognizes MIX Xtras (although most are inappropriate for SW).

Shockwave for Macintosh Xtras folder

In SW7 for Macintosh, all Xtras are installed under *System Folder:Extensions:Macromedia:Shockwave:Xtras*. For Netscape on the Macintosh, in Shockwave 6, the

support folder has the same name as the plug-in with a space and the word "folder" appended. For example, the SW6 Xtras folder would be called:

Netscape Navigator™ Folder:Plug-ins:NP-PPC-Dir-Shockwave folder:Xtras

or:

Netscape Navigator™ Folder:Plug-ins:NP-68K-Dir-Shockwave folder:Xtras

Shockwave for Windows Xtras folder

In SW7, all Xtras are installed under *C:\Windows\System\Macromed\Shockwave\Xtras*. In SW6, for Netscape on Windows, the support folder has the same name as the plug-in (without the .DLL extension). The 32-bit plug-in name is *NPDSW32.DLL*, so Xtras go under *Netscape\Plugins\NPDSW32\Xtras*.

In SW6, for Microsoft Internet Explorer on Windows 95, the support folder is the entire *Windows\System* directory, but you can place Xtras in *Windows\System\Xtras* to keep them separated from other files.

Including Xtras with a Projector

When using Xtras with Projectors, you can place them in a Projector-specific *Xtras* folder, bundle them into the Projector, or use some combination of the two. I strongly suggest leaving the Projector's Xtras exclusively in an *Xtras* folder.

Which Xtras to Distribute

Director 6 and 7 use an ungodly number of Xtras. Tables 10-2 through 10-5 and 15-13 list Xtras needed for various operations, but it is still easy to be confused about which Xtras your Projector requires.

The Xtras to distribute are sometimes the same ones as used during authoring, but not always (as with the QT3, Flash, and Custom Cursor Asset Xtras). There are also many Xtras that are needed only at authoring, such as Tool Xtras. Never distribute Xtras meant only for authoring-time such as the agent Xtras or the PowerPoint Import or Java Export Xtras.

See *http://www. zeusprod.com/nutshell/appendices/checklists.html* for a list of Xtras and other files to be included with the Projector.

To include the list of Xtras under `Modify ➤ Movie ➤ Xtras` automatically in your Projector, check the *Check Movie for Xtras* checkbox when creating your D6 Projector. In D7, each Xtra in the list can be included individually. I prefer to peruse the list and manually include those Xtras in my Projector's *Xtras* folder. In D7, check *the movieXtraList* property.

Network Xtras

If you use any linked cast members, sounds, movies, or castLibs at a URL, you'll need the INetURL and NetFile (and probably NetLingo) Xtras. If using linked GIF and JPEG images, you'll also need the GIF Import and JPEG Import Xtras. When using a PowerPC-native or Fat Projector on a PowerMac, you'll also need the NetManage WinSock Lib file. The D6 *Include Network Xtras* checkbox under `File`

➤ `Create Projector` ➤ `Options` will include the Network Xtras whether needed or not. In D7, Network Xtras are added under `Modify` ➤ `Movie` ➤ `Xtras`. See Table 10-2 for important details.

You'll need the NetLingo Xtra if using any of the Lingo commands shown in Table 11-5. The NetLingo commands are built into the Shockwave plug-in and do not require an Xtra when running under Shockwave. Thus, not every command that requires an Xtra during authoring requires one in Shockwave.

MIX and MIX Services Xtras

The Xtras needed when linking to external graphics are shown in Table 10-3. These assets are typically added via `File` ➤ `Import` using the *Link to External File* option.

You do not need any MIX Xtras at runtime if using only internal (embedded) assets. The `Modify` ➤ `Movie` ➤ `Xtras` option should show all the MIX Xtras you need (plus the MIX Services Xtra) automatically. Director updates this list of Xtras whenever you use an externally linked asset. If you have deleted some linked cast members, the list may show Xtras that are no longer needed. If you remove Xtras from the list, Director will re-add them automatically if they are still needed.

For example, if you link to any external sounds (AIF or WAVE files) or use *sound playFile*, you should include the Sound Import Export Xtras and the MIX Services Xtra.

Sprite Asset Xtras and Transition Xtras

If you insert assets that requires Sprite Xtras, include the Xtras with your Projector, as shown in Tables 10-4 or 10-5. Sprite assets are added via `File` ➤ `Import` in D7 or the `Insert` ➤ `Media Element` or `Insert` ➤ `Control` menu in D6 and D7. These include Custom Cursors, Active X controls, QuickTime 3 media, Shockwave audio, and Flash files, plus the new types in Table 10-5. As with MIX Xtras, Director adds the necessary Xtras to the list under `Modify` ➤ `Movie` ➤ `Xtras` automatically whenever you insert a new cast member type that requires an Xtra.

The Flash Asset Xtra is included in the Shockwave for Director download, so you don't ordinarily need to include it in Shockwave projects (it is also needed for D7 vector shapes).

If you use any third-party Sprite Xtras, include them with the Projector, too. Director's built-in Transitions do not require Xtras, but third-party Transitions do.

You do not need Xtras for QuickTime 2 or AVI *#digitalVideo* cast members, or for embedded bitmaps or sounds. You will need a Sprite Xtra for QuickTime 3 cast members.

Lingo Xtras

Lingo Xtras don't show up in the list under `Modify` ➤ `Movie` ➤ `Xtras` unless you add them manually. You need to be aware of which Lingo Xtras you are using and be sure to include them with your Projector. Some are shown in Table 10-4, but you will likely use third-party Xtras as well.

If you omit an Xtra, such as the NetLingo or FileIO Xtra, and try to use a command that requires that Xtra, Director will display a "Handler not defined" error.

You can check the list of commands contained within an Xtra using the Message window, such as:

```
put mMessageList (xtra "NetLingo")
```

or:

```
put interface (xtra "NetLingo")
```

See Chapter 13 in *Lingo in a Nutshell* for additional details on Lingo Xtras.

The Xtras Folder for Projectors

Projectors will load Xtras from the folder called *Xtras* under the folder from which the Projector is running. This allows each Projector to use different Xtras.

Copy the distributable runtime versions of the Xtras, which often differ from the authoring versions used during development, to the Projector's *Xtras* folder.

Xtras Bundled into the Projector

Director 6 and 7 allow you to bundle Xtras into a Projector; Director 5 does not. I see no compelling reason to bundle Xtras, especially because Projectors unbundle their Xtras into a temporary folder at runtime. On writable media, a temporary *Xtras* folder for unbundled Xtras is created within the folder where the Projector resides. For read-only media, a temporary folder on the system disk, as shown in Table 10-6, is used instead.

Reasons not to bundle Xtras:

- Xtras are unbundled at runtime, which slows Projector startup by 15 seconds or more.
- If run from a locked volume (CD-ROM), Xtras are copied onto the user's hard drive, requiring hard disk space and slowing performance. This will fail on systems that prohibit write access to the Windows system folder.
- Xtras could be left behind in the temporary folder if the Projector crashes (the related bug was fixed in D6.0.2).
- Automatic bundling, via `File` ➤ `Create Projector` ➤ *Options* checkboxes in D6 or via `Modify` ➤ `Movie` ➤ `Xtras` in D7, includes unnecessary Xtras in some cases. It is better to select exactly the Xtras you need.
- Automatic bundling doesn't necessarily include all needed Xtras, such as Lingo Xtras and the NetFile Xtra needed for local SWA playback.
- Bundling Xtras forces you to rebuild your Projector to add, test, or remove different Xtras.
- Bundling Xtras has been reported to prevent Director from reading Xtras from the *Xtras* folder, but this is unconfirmed.
- Projectors with bundled Xtras cause the error: "Problem opening C:\WINNT," or a similar error under Windows NT if the drive is larger than 4 GB.

- Bundled Xtras are not supported in Shockwave or prior to D6.
- Slim Projectors in D7 use the Xtras in the SW7 *Xtras* folder. Bundling the default Xtras is largely redundant.

If you insist on bundling Xtras, there are several methods:

- Add Xtras to the Projector's build list manually via the file picker when creating the Projector using `File➤Create Projector`.
- In D6, use the *Check Movie for Xtras* checkbox under `File➤Create Projector➤` *Options*. It interacts with the `Modify➤Movie➤Xtras` option and the *XTRAINFO.TXT* file. Use the *Include Network Xtras* checkbox under `File➤Create Projector➤` *Options* in tandem with the *XTRAINFO.TXT* file.
- In D7, use the *Include in Projector* checkbox for individual Xtras under `Modify➤Movie➤Xtras`.

Automatic bundling includes only those Xtras used by movies embedded in the Projector (although other Xtras can be added to the Projector build list). I recommend using a Stub Projector containing only one minimal movie.

The Modify➤Movie➤Xtras options

When you add a cast member that requires a Sprite, Transition, or MIX Xtra, the appropriate Xtras are added automatically to the *Movie Xtras* dialog box accessed via `Modify➤Movie➤Xtras` (see Figure 10-1). The Xtras list doesn't include Lingo Xtras unless you add them manually. If you remove an Xtra that is needed, Director will reinstate it the next time you use the menu option to view the list. If you delete all cast members that use a Sprite Xtra, perform a `File➤Save and Compact` to purge the movie of the old assets, then delete the Sprite Xtra from the list manually, as Director won't do this automatically.

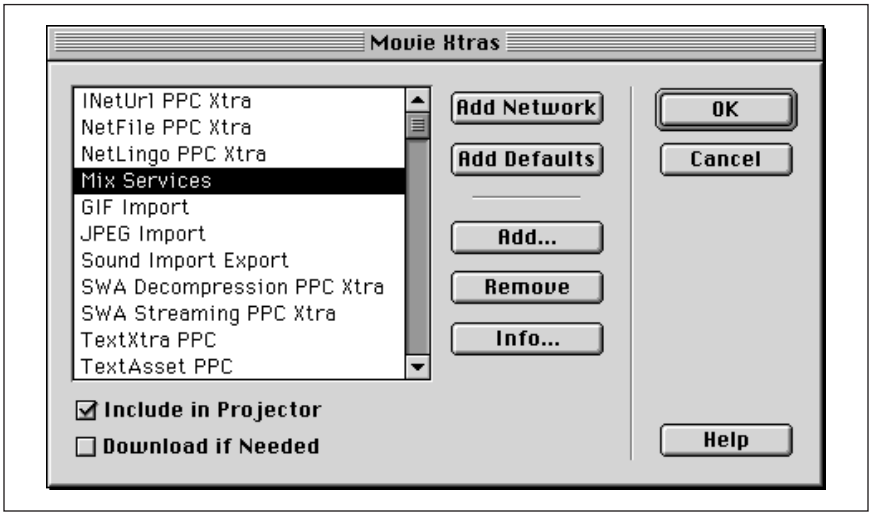


Figure 10-1: Movie Xtras dialog box

In D6, items in the Modify ➤ Movie ➤ Xtras list are bundled with the Projector if the *Check Movie for Xtras* checkbox is checked under File ➤ Create Projector ➤ Options. (Only those Xtras specified for movies included in the Projector are bundled. Xtras for external movies are not.) This checkbox also adds Xtras of type *#mixin* from the *XTRAINFO.TXT* file in the Projector.

Rather than bundle Xtras into the Projector, I use the Xtras list to help determine which Sprite and MIX Xtras to include in the *Xtras* folder.

You can use the *Add* button under Modify ➤ Movie ➤ Xtras to see a list of many (but not all) of the Xtras installed, including Lingo, Sprite, and MIX Xtras. The D7 property *the xtraList* contains all installed Xtras.

In D7, the *Add Defaults* button adds Xtras flagged as `#type:#default` and the *Add Network* button adds Xtras flagged as `#net:#xtra` in the *XTRAINFO.TXT* file. I prefer to delete the *#default* flag from the *XTRAINFO.TXT* entries so that unnecessary Xtras are not listed for each movie.

The list of Xtras is stored in each DIR file (Director movie). If (and only if) there is a corresponding entry in the *XTRAINFO.TXT* file, the Xtra names under Modify ➤ Movie ➤ Xtras will be translated to the appropriate platform as needed. In D7, the property *the movieXtraList* contains the Xtras listed there.

When the movie is opened from a Projector or during authoring, this list of Xtras is checked, and an alert dialog box is shown for any missing Xtras. (This dialog box cannot be suppressed via *the alerthook*.) If you delete needed Xtras from the Xtras list, but never reopen the Xtras list via Modify ➤ Movie ➤ Xtras, they will not be added back to the list. This prevents an error dialog box from appearing when the movie is run in a Projector, even if the needed Sprite Xtras are missing. During authoring, you are likely to get an error message.

In D7, the *Download if Needed* checkbox and *Info* button are available only if the *XTRAINFO.TXT* file includes the `#package` and `#info` flags for the Xtra. You may experience a long delay or crash if a network connection is not active when using these options.

Detecting Installed Xtras

In D7, the new property *the xtraList* contains a list of property lists showing the name and version of installed Xtras, such as:

```
[[#name:"FileXtra", #version:""], [#name:"UIHelperPPC Xtra",
#version: "7.0"]...]
```

The *showXlib* command lists only Lingo Xtras in the Message window. At runtime, you can use *the netPresent* to determine whether the *NetLingo* and *NetFile* Xtras are installed (it doesn't check for the *INetURL* Xtra). See Examples 13-6 and 13-7 in Chapter 13 in *Lingo in a Nutshell* for a utility that determines which Lingo Scripting Xtras are installed using the *number of Xtra* and *name of Xtra* properties.

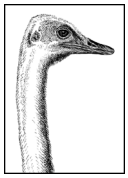
See also the list of Xtras displayed via Modify ➤ Movie ➤ Xtras ➤ Add, although it is incomplete in D6. In D6, to determine whether an Asset Xtra is installed, you can attempt to create a new cast member and check the result, as shown in Example 10-1. In D7, you could check *the xtraList* instead.

Example 10-1: Checking Whether an Asset Xtra Is Installed

```
on assetXtraInstalled assetType
  -- Try to create a new cast member
  set asset = new (assetType)
  if ilk(asset, #member) then
    -- Erase it when done
    erase asset
    return TRUE
  else
    return FALSE
  end if
end assetXtraInstalled
```

The following code checks whether the Flash Sprite Asset Xtra is installed:

```
put assetXtraInstalled(#flash)
-- 1
```



CHAPTER 15

Sound and Cue Points

Director is not a sound editing tool. You will usually create your sound files in an external application, and then import them into Director. For testing, you can use the sample sound files on the Director 7 CD in the *Macromedia/Support* folder.

D7 includes the Beatnik Lite Xtra (see `Xtras ➤ Beatnik Lite` or *http://www.headspace.com/to/?xtra0support*). The D7 Studio for Macintosh includes Bias Peak LE (see *http://www.bias-inc.com*) instead of SoundEdit, which was bundled with previous Studio versions. The D7 Studio for Windows includes Sonic Foundry's Sound Forge XP, as did the D6 Studio.

Digital Audio Primer

A brief primer on digital audio is in order. In the real world, sound is continuous. For computer use, sound is digitized by sampling it at many points throughout each second of audio. A sound's *sample rate* is typically 11.025, 22.050, or 44.100 kHz (kilohertz, or thousands of samples per second), although many variations exist. Higher sampling rates require more storage, but typically sound better (have higher fidelity). The *bits per sample* or *sound bit depth* is usually 8 or 16, meaning that each sample requires 1 or 2 bytes of storage (there are 8 bits per byte).

Downsampling refers to lowering a sound's sample rate or bit depth, which can be done in a sound editor, but not in Director. Downsampled audio uses less storage at the expense of quality. *Resampling* is the process of changing the sampling rate, either up or down. The fidelity of resampled audio can vary measurably between different sound editors.

Sounds typically contain one or two channels: mono (or monaural) or stereo. To *fold* or *flatten* a sound is to convert it from stereo to mono, sometimes at the expense of quality.

The size of an uncompressed sound can be determined as:

```
size in K = (bits per sample)/8 × (sample rate in kHz) ×
     (number of channels) × (duration in seconds)
```

For example, 22.050 kHz, 16-bit mono sound occupies:

```
16/8 × 22.050 × 1 = 44.1 K/sec
```

And 44.100 kHz, 16-bit stereo sound occupies:

```
16/8 × 44.100 × 2 = 176.4 K/sec
```

For compressed Shockwave audio (SWA), the important factor is the streaming data rate, which is chosen when you compress your SWA, measured in Kbps (thousands of bits per second):

```
(bit rate in Kbps)/8 = data rate in K/sec
```

Audio data is typically measured in K/sec. Memory is typically measured in increments of 1024 bytes (KB). Throughout this chapter, I use *K* to indicate 1000, and *KB* to indicate 1024. For example, CD-quality audio (16-bit, 44.1 kHz, stereo) requires 176.4 K/sec, which is technically 172.27 KB/sec. Divide by 1.024 to convert from K to KB, and divide by 8.192 rather than by 8 to convert from Kbps to KB, to account for the 2% discrepancy.

To calculate the disk space required for any type of sound, multiply the rate in KB/sec by the duration in seconds. These calculations exclude the small header associated with each sound. Unless the sound is very short, has a low data rate, and contains numerous cue points, the header size is insignificant relative to the audio data.

Sound Playback in Director

Director uses two Sound channels in the Score that correspond to the first and second system sound channels. But SWA sprites, which are placed in the sprite channels, can also use system sound channels 1 and 2. I use the uppercase *Sound channel* when referring to the channels in the Score, and the lowercase *sound channel* when referring to the System sound channels accessible via Lingo.

See "Xtras needed to play external sounds in Director 6 and 7" later in this chapter to ensure that you've included the necessary Xtras with your Projector.

Supported Sound Formats

Sound cast members may be imported (embedded) into Director's cast or may be externally linked. Table 15-1 lists the supported audio formats as well as cross-platform differences. Note that the same AIFF, WAVE, and SWA files can be used on both Macintosh and Windows in D6 and D7. QuickTime and AVI files can also contain audio tracks. The Java player supports Sun AU format only.

Director does not export sound files, and sounds may be lost when exporting to QuickTime or AVI formats. To export a sound, cut and paste from the Cast to your sound editing program, set an external sound editor under `File` ➤ `Preferences` ➤

Editor, or rely on a third-party Xtra. There should be no need to export sounds in most cases, as you should retain the original source files imported into Director.

Table 15-1: Cross-Platform Audio Comparison

Feature	Macintosh	Windows
Formats supported for import into Cast[1]	SWA, AIFF, AIFC,[2] WAVE, System 7 sounds, Sun AU,[3] MPEG3	Same as Mac, except for System 7 sounds
Formats supported for external linking	Same as above (except System 7 sounds), plus QuickTime (AVI imported as QT3 only)	Same as above, plus QuickTime, AVI
Supported sampling rates[4]	5.564, 7.418, 11.025, 11.127, 22.050, 22.255, 32.000, 44.100, and 48.000 MHz	11.025, 22.050, and 44.100 MHz
Multichannel audio	Built-in (zero latency)	Supported via *MacroMix*, DirectSound, or *QT3Mix*[5] (non-zero latency)
Maximum number of sound channels	8	Up to 8 (set by *MixMaxChannels* in *DIRECTOR.INI*)
Multiple simultaneous audio sources	Yes	Not necessarily[5]
System audio buffer size[6]	System's audio buffer size is fixed	Settable via *DIRECTOR.INI* file.
the soundLevel (0 to 7)	Matches settings in *Sound* or *Monitors & Sound* Control Panel	Matches *SoundLevel* settings in *DIRECTOR.INI*. See also *Volume Control* in Task Bar[7]

[1] Any internal sounds are stored in Director's internal format, and are completely cross-platform. D7 can import SWA files via File ➤ Import, but they are converted to Director's internal format.
[2] AIFC is an AIFF file with IMA compression.
[3] The Sun AU Import Export Xtra included with D6.5 and D7 is required for Sun AU file support.
[4] Unsupported audio sampling rates under Windows are resampled to the nearest supported rate on the fly, distorting the pitch in some cases. The Macintosh supports just about any sampling rate, including variants such as 11,126, 22.253, and 22.254 MHz.
[5] Windows does not necessarily allow Director-based and QuickTime or AVI-based audio to be played simultaneously and multiple sounds introduce latency. See "Sound Mixing Under Windows" later in this chapter for details.
[6] The system audio buffer size is unrelated to the SWA buffer length, which can be set for each SWA member.
[7] The Windows Volume Control accessory can be opened by double-clicking the speaker icon in the Task Bar or running *C:\Windows\SNDVOL32.EXE*, which is accessed via Start ➤ Programs ➤ Accessories ➤ Multimedia ➤ Volume Control.

SoundEdit's native SE16 format was supported by D5, but is not supported in D6 or D7. Likewise, sounds compressed with MACE 3:1 and MACE 6:1 are not supported. D6 and D7 support IMA-compressed WAVE files, but not WAVE files compressed with Microsoft's proprietary compression. Sun AU files must have the .au extension (the equivalent Mac file type, ULAW, was not recognized prior to D7.0.1). See Table 4-4.

User Interface Issues

High-quality audio enhances a multimedia experience more than you might realize. When designing your audio, keep the following in mind:

- Director will automatically continue playing sounds at the end of one movie when branching to another movie. This provides an audio "transition."
- Use professional-quality sound effects and voice-overs (sound effects collections and professional voice talent are widely available).
- Avoid loud, annoying sound loops.
- Not all computers have a sound card and speakers, or the volume may be muted (and some users are deaf).
- Include text prompts for vital operations and allow the user to replay important audio messages.
- Sound cards and speaker quality vary. Not all systems support CD-quality stereo sound. The highest quality sounds may be a waste of bandwidth.
- Provide a volume control with a mute option. It is exceedingly rude to increase the system volume level automatically. If necessary, check the volume level via Lingo and suggest that the user change it.

Comparison of Sound Playback Methods

The various sound playback options are shown in Tables 15-2 and 15-3. The optimal method depends on the playback platform, the format of the sound, the size of the sound, the number of sound channels in use, the presence or absence of simultaneous animation and video, and whether a sound is triggered by an event, used as background music, or played in synchronization with a video or animation.

Table 15-2: Audio Playback Method Comparison

Method	**Pros**	**Cons**
Score Sound channel	Easy and intuitive. No scripting. Can use Tempo channel to wait for sounds.	Limited control and poor synchronization; only two sound channels.
puppetSound	Control over sound triggering.	Sounds must be explicitly unpuppeted.
sound playFile	Sounds don't require a cast member. Can be played in any sound channel.	Slows loading of video or graphic media.
Sound tracks in digital video cast member	Can start and stop sound at any point, or play sound fast, slow, or backward.	QT and AVI audio may not mix with other audio under Windows
SWA	High quality at low bandwidth (good compression). Accesses up to 8 sound channels.	Processor-intensive. Requires 68K with FPU, or Pentium for playback.
Flash Audio	Compact and integrated with Flash file.	Low quality. Conflicts with Director sounds.

Table 15-2: Audio Playback Method Comparison (continued)

Method	Pros	Cons
MIDI	Small size with excellent fidelity. Instruments can be changed dynamically.	Not well supported. Requires QT with MIDI extension or third-party Xtra.
MCI	Control over external devices, such as video disks and RedBook CDs.	Windows-only. Not universally reliable.
Third-party Xtras	The free CDPro Xtra (*http://www.penworks.com*) plays RedBook audio. Beatnik adds many capabilities.	Beatnik Xtra is expensive.

Table 15-3 compares features of the various playback methods. The Macintosh supports up to eight sound channels. The number of channels on the PC is set by the *MixMaxChannels* option in the *DIRECTOR.INI* file.

Table 15-3: Audio Playback Method Features

Method	Streamed	Loopable[1]	Channels	Cue Points
Score Sound channels	Only if external	Only if internal	2	AIFF, and WAVE (D6.5 or later)
puppetSound	Only if external	Only if internal	Up to 8	Same as above
sound playFile	Always	No	Up to 8	Same as above
Audio track in digital video	Always	No	Uses separate video mixer	Same as above
SWA	Only if external	Only if internal	Up to 8	Yes, for QT and AVI
Flash audio	N/A	No	N/A	No
MIDI	Varies	Yes	N/A	No
Third-party	Usually	Varies	Varies	Xtra-dependent[2]

[1] Internal sounds are generally loopable, whereas external sound files are not. Even sounds that are not automatically loopable can be looped manually via Lingo, although the loop may not be as seamless as with internal sound cast members.
[2] The third-party MPEG Xtra supports cue points, as may other Xtras.

Sound Cast Members

Sound cast members may be imported (embedded) into Director's Cast or externally linked (see "Import Options: To Link or not To Link" in Chapter 4). Import short and frequently used sounds into the Cast. Leave longer sound files on disk and link to them instead. See Example 4-7, which imports small external sounds into the Cast and warns about large internal sounds.

Standard sound and SWA cast members are indicated by a speaker icon (see Figure 4-3). Find sound cast members under `Edit ➤ Find ➤ Cast Member` by searching for members of *Type: Sound*; find SWA members using *Type: Xtras* in D6

and *Type: Shockwave Audio* in D7. Find QT2 or AVI members (which often contain audio) using *Type:Digital Video*. Find QT3 members using *Type:Xtra* in D6.5 and *Type:QuickTime 3* in D7.

External sound files can be changed without altering your Director movie and allow you to easily ship audio in different languages or audio of varying quality. Similar flexibility can be achieved by using an externally linked castLib to hold your internal sound cast members.

During development, either of these techniques prevents your main movie from growing large due to the inclusion of sounds. As sounds don't often change, this allows you to transfer, back up, or distribute only those portions of the project that have been altered.

Internal (embedded) sounds

Embed small sounds (those under approximately 500 KB) in the Cast using the *Standard Import* mode under `File ➤ Import`. An internal sound is preloaded in its entirety before it plays (or can be preloaded manually ahead of time), so any disk access occurs before, not during, playback. Internal sounds remain in memory after being played (until being unloaded when Director needs the RAM), and need not be reloaded each time they are played. Thus, internal sounds are convenient for button-click noises and small looping sounds. These same attributes make embedded sounds inappropriate for large sounds; large internal sounds cause long load delays and consume excessive memory.

Externally linked (streaming) sounds

Link to large AIFF and WAVE sounds using the *Link to File* mode under `File ➤ Import`. Externally linked sounds are streamed from disk and begin to play as soon as the first data is available. A streamed sound can be of arbitrary length without requiring significant RAM. Linked sounds are most appropriate for long sounds used only once, such as narration.

However, streamed sounds are not kept in RAM, cannot be looped automatically, and must be reloaded to be repeated. Because a CD-ROM can't access data from two places simultaneously, streaming audio will hinder the loading of other data such as digital video or bitmaps. The key factor for external streaming sounds, including SWA, is the *bandwidth*, not the total file size; as Buzz Kettles puts it, "It's not the size, it's the motion."

When using linked sounds, include the external sound files and sound-related Xtras with your Projector (see "Sound-Related Xtras" later in this chapter).

Sounds played via *sound playFile* behave similarly to linked sounds, although they need not be imported at all.

The path to external sounds as indicated by *the fileName* or *streamName of member* updates automatically for the current platform if the sound was imported via `File ➤ Import` or inserted via `Insert ➤ Media Element ➤ Shockwave Audio` in D7. Assuming that the external sound file remains in the same position relative to the movie or castLib and that *the fileName* or *streamName* is valid for the current platform, Director 7 will find the external audio file. However, the path

to SWA files inserted via `Insert ➤ Media Element ➤ Shockwave Audio` in D6 (as indicated by *the streamName of member*) will not update automatically when files are moved, even if the same relative positions are maintained. See "Common importing and linked file problems" in Chapter 4 for the solution.

Place external sound files close to the appropriate Director movie when burning a CD to reduce the seek time (and latency) for accessing external sounds.

Differentiating between a sound's type and the playback method

In most cases, a sound's format or cast member type dictates whether it will be internal or external, but a standard sound cast member can be either. However, a cast member's or sprite's type (*#sound*, *#SWA*, *#digitalVideo*, or *#quickTimeMedia*) will determine which commands support it. For example, *puppetSound* and the Score Sound channels support only *#sound* cast members (linked or unlinked). See Table 15-4 for a comparison of Lingo commands used with the different sound formats.

So-called "streaming" sounds are not necessarily located on a remote server. All external sound files are streamed, whether from the Internet, CD-ROM, or hard drive. Likewise, although SWA compression is usually associated with streamed Internet audio, SWA sounds can be streamed from a local drive, and internal Director sounds can be SWA-compressed, in which case they are not streamed.

Sound Playback Methods

Your presentation will probably use more than one method to play sounds. They can be combined, subject to these limitations:

Sound channel conflicts
: Each sound channel can play only one sound at any time. You can't use Sound channel 1 in the Score while simultaneously using *sound playFile* or *puppetSound* to also play a sound in channel 1.

Sound device or sound driver conflicts
: Only one device or driver can access the sound card at any given time. Under Windows, because QuickTime and Director audio often play through different mechanisms, the two types of audio cannot always be combined. See "Sound Mixing Under Windows" later in this chapter. The Macintosh handles multiple sound sources transparently, but even there, Director can not access the data track of a CD while a RedBook audio Xtra is accessing RedBook audio tracks on the same CD.

Sound in the Score

Sound cast members are placed in one of the two Sound channels of the Score. SWA and digital video (DV) cast members, including audio-only QuickTime movies, are placed in sprite channels, not the Sound channels.

Score-based sounds do not play unless the Score's playback head is moving or looping in a frame (avoid the *pause* command). Because Director's frame rate is not exact, you should use cue points to synchronize sounds with Score animations.

Sound playback is not affected by the Tempo channel's frame rate setting or the *puppetTempo* command. Standard and SWA sounds always play back in real time, although the speed of a DV sprite, including the audio tracks, can be changed via *the movieRate of sprite*. (Slower playback lowers the pitch of the audio track; faster playback raises the pitch.)

Sounds in the Score Sound channels

Score Sound channels are best used for fixed sounds that accompany animations, but the triggering and synchronization of Score sounds is insufficient for lip-synching or other time-critical uses.

Sounds in the Score are triggered when a sound is first encountered in a Sound channel or an SWA or DV sprite span is encountered in a sprite channel. Sounds ordinarily terminate after playing once through, even if they are tweened out over additional frames. To re-trigger a sound, you must either use Lingo or create a break of at least one empty frame in the Sound channel. Internal sounds that are looped will play as long as they occupy a sound channel or until the Tempo channel's *Wait for Cue Point: End* option causes them to "play out."

puppetSound

Use *puppetSound* to trigger a sound cast member (whether linked or unlinked) at an arbitrary time, such as in response to button-clicks or timeouts. *PuppetSound*s played in channels 1 and 2 override the corresponding Score Sound channel. Use:

```
puppetSound channel, "soundMemberName"
puppetSound channel, member "whichMember" {of castLib whichCast}
```

where `channel` is a number from 1 to 8. If `channel` is specified, the sound will trigger immediately. Otherwise, the default channel is 1, and the *puppetSound* is not triggered until the playback head advances (or loops) or an *updateStage* command is issued. Here, the default channel 1 is assumed:

```
puppetSound "soundMemberName"
puppetSound member "whichMember" {of castLib whichCast}
```

Specifying 0 as the "sound" unpuppets a *puppetSound*. Unpuppetting halts the current *puppetSound*; unpuppetting channel 1 or 2 also returns control to the corresponding Score Sound channel. These commands unpuppet a sound channel:

```
puppetSound channel, 0
puppetSound 0 -- Unpuppets sounds channel 1 by default
```

Sound playFile

The *sound playFile* command streams an external AIFF, AIFC, or WAVE file from disk, similar to a linked sound cast member. The sound need not be imported into the Cast nor appear in the Score, but it must reside on disk. *Sound playFile* can play a file at a remote URL, but it does not support SWA files, so you should use linked SWA cast members for net-based sounds. If you choose to play a remote AIFF or WAVE, the file should be downloaded first with *downloadNetThing*.

Using *sound playFile* with channel 1 or 2 overrides the corresponding Score Sound channel, but control automatically returns to the Score when the sound terminates. *Sound playFile* even overrides puppeted sound channels. *Sound playFile* assumes that the external sound file has an .AIF extension, if none is specified. It takes the form:

```
sound playFile channel, soundFilePath | url
```

where *channel* is from 1 to 8. If the *channel* is omitted, channel 1 is assumed:

```
sound playFile 1, the moviePath & "mysound.aif"
sound playFile 3, "http://www.zeusprod.com/examples/sound.wav"
```

The path to the external file specified by *sound playFile* does not update automatically. You must manually specify the path on each platform, as shown in Example 15-1, or use the @ operator to create a generalized path. Note that we constructed a path relative to the current movie's location.

Example 15-1: Specifying a Path to a Sound

```
if the platform contains "Windows" then
  sound playFile 1, the moviePath & "audio/mysound.aif"
else
  sound playFile 1, the moviePath & "audio:mysound.aif"
end if
```

Example 15-2 shows how to construct a central convenience function to play voice-overs from a subfolder named *VO*, and ambient sounds from a subfolder named *AMBIENT*, each below the folder containing the current movie.

Example 15-2: Centralized Sound PlayFile Commands

```
on playVoiceOver someSound
  -- Play voice-overs in channel 1
  set pathSeparator = the last char of the moviePath
  sound playFile 1, the moviePath & "VO" & ¬
     pathSeparator & someSound & ".AIF"
end playVoiceOver

on playAmbient someSound
  -- Play ambient sounds in channel 2
  set pathSeparator = the last char of the moviePath
  sound playFile 2, the moviePath & "AMBIENT" & ¬
     pathSeparator & someSound  & ".AIF"
end playAmbient
```

In Example 15-2, the sound files are assumed to have an .AIF extension and reside in the *VO* or *AMBIENT* subfolder. When using these routines, we can specify sound filenames without the .AIF extension or the folder name, such as:

```
playVoiceOver ("intro")
playAmbient ("mood")
```

As with all external sound playback, *sound playFile* is appropriate for long sounds. Such sounds won't automatically loop; you must use *sound playFile* again to retrigger them.

Digital video sounds

Digital video (DV) cast members are played in sprite channels, not the Sound Score channels, and may contain audio tracks (*#sound* or *#midi*) even if no *#video* track is present. DV (QT or AVI) provides better synchronization than other audio playback methods, provided that the audio and video being synchronized are properly interleaved in a single DV file. See Chapter 16, *Digital Video*, for complete details.

DV sprites, including their audio tracks, are played automatically when a DV sprite is encountered, provided that *the pausedAtStart of member* is `FALSE`. Lingo commands can start, stop, rewind, or fast forward to any point in the DV file. DV files, which are always external, can be played at different speeds or even backward. But DV audio may not be playable simultaneously with other standard Director sounds under Windows. See "Sound Mixing Under Windows" later in this chapter.

Shockwave Audio (SWA)

SWA cast members are played in sprite channels, not the Sound Score channels. Shockwave audio offers high quality at low bandwidths, but requires more processing power at runtime and isn't supported on some low-end machines. (SWA compression requires a Pentium or PowerMac.) SWA requires several Xtras (see Table 15-13) that must be distributed with your Projector. If using SWA from Shockwave 6 or 7, the Xtras are included with Shockwave browser plug-in.

Flash audio

Audio can be added to a Flash cast member prior to import into Director, but Flash-based audio is low-fidelity and may conflict with Director. (Flash-based audio and Director-based audio can't play at the same time under Windows.) Instead, trigger Director-based sounds from your Flash sprite, offering Lingo control over volume, better integration with other Director sounds, and optional SWA compression.

Sound Operation Comparison

Table 15-4 compares the inconsistent sound control commands across the different sound-related cast member types. Digital Video includes AVI and QuickTime prior to version 3 (QuickTime 2.5 on Macintosh or QuickTime 2.1.2 under Windows). QuickTime 3 requires the new QT3 Asset Xtra in D6.5 or D7. See also Chapter 16, *the digitalVideoType of member*, and Table 4-10.

External files played via *sound playFile* do not support any sprite or member properties. They obey only the *SoundBusy(), sound level*, *sound stop*, and *sound close* commands.

Table 15-4: Common Sound Operations

Operation	Sound	SWA or MP3	QuickTime 3 or Digital Video
Type of member	#sound	#SWA	#digitalVideo or #quickTimeMedia
Number of channels in asset	channelCount of member	numChannels of member	trackCount(), trackType = #sound
Specify sound channel	Place in Score or puppetSound or sound playFile	soundChannel of member (0 assigns highest)	Allocated by OS
Volume[1]	volume of sound	volume of member, volume of sprite	volume of sprite, sound of member, volumeLevel of sprite
Play in Score	Sound channel	sprite channel	sprite channel
Play via Lingo	sound playFile, puppetSound	play ()	movieRate of sprite, mRate of sprite[3]
Stop a sound	sound stop, puppetSound 0, sound close	stop(), pause()	set movieRate of sprite = 0, or set mRate of sprite = 0
Sound playing	soundBusy()	state of member = 3	movieRate of sprite, mRate of sprite, pausedAtStart of member
Wait for sound[2]	soundBusy()	state of member	movieRate, movieTime, mRate, or mTime of sprite[3]
Samples per second	sampleRate of member	sampleRate of member, bitRate of member	Not available
Bits per sample	sampleSize of member	bitsPerSample of member	Not available
Current position	currentTime of sound	percentPlayed, percentStreamed, and currentTime of sprite	movieTime, mTime, or currentTime of sprite
Length in seconds	See Example 15-3	duration of member × 1000	duration of member × the digitalVideoTimeScale

[1] See "Volume Levels and Sound Fades" later in this chapter.
[2] You can use the Tempo channel or cue points to wait for any type of media that supports them. See "Cue Points and Timing" later in this chapter.
[3] In D6.5, use *the volumeLevel*, *mRate*, and *mTime of sprite* properties for QT3 sprites. In D7, these properties are deprecated; use *the volume*, *movieRate*, and *movieTime of sprite* instead.

There is no *duration of member* property for standard sound cast members. Example 15-3 calculates the duration of a sound based on its other attributes. The *size of member* property is accurate only for internal (embedded) sounds, but can be calculated for external sounds (see Examples 4-6 and 4-7).

Example 15-3: Calculating the Duration of an Internal Sound

```
on soundDuration whichSound
  -- Returns the duration in seconds
  set duration = the size of member whichSound / ¬
        (the sampleRate   of member whichSound * ¬
         the channelCount of member whichSound * ¬
         the sampleSize   of member whichSound  / 8.0)
  return duration
end soundDuration
```

Sound Channels and Sound Mixing

Director supports up to eight sound channels, although only two are shown in the Score. The remainder are accessible via *puppetSound* or *sound playFile* or are used implicitly by SWA sprites, film loops, and MIAWs (which all share the same sound channels). Digital video and Flash sprites do not use the same sound channel numbers as standard Director sounds.

Director's Score Sound channels do not correspond to the left and right channels of a typical stereo. Monaural sounds are split equally between the left and right speakers regardless of the Director sound channel used. A so-called *stereo* sound uses only one of Director's sound channels because the left and right audio tracks are interleaved into a single data stream (think of Director's sound channels as *data* channels, rather than *audio* channels). The sound card knows how to split a stereo data stream between the right and left speakers, but a sound's left/right balance can not be set in Director without an Xtra (or you change the balance using an external sound editor before importing into Director).

Avoid sound channel number conflicts by simply specifying different channel numbers for *puppetSounds* or *sound playFile* commands (both use channel 1 by default, which will override the Score's Sound channel 1). If *the soundChannel of member* of an SWA member is 0 (the default) it automatically sidesteps conflicts by using the highest available channel number.

Most Macintoshes support eight sound channels, although older Performas may support only four. On the Macintosh, all sound playback is handled seamlessly by the Sound Manager, which is a QuickTime component. You can play almost any type or number of sounds without regard to conflicts at the Mac OS level.

The number of audio channels supported under Windows is generally at least four, but even assuming that you have not used conflicting channel numbers, other audio sources (digital video and Flash) can cause conflicts at the so-called Windows device level.

Sound Mixing Under Windows

Windows PCs have only one *hardware* sound channel. Multichannel sound is simulated by premixing multiple audio sources before sending the resultant audio stream to the sound card. There are a number of competing and complementary sound drivers, mixers, and devices that ameliorate the latency and conflicts arising from playing multiple sounds under Windows. The best method and achievable results vary with the software configuration, Windows version, and Director

version. For an overview of sound mixing under Windows, see Macromedia TechNote #03191, "Windows and Multichannel Sound."

Sound mixing latency

Playing two or more sounds simultaneously under Windows may cause a delay as the sounds are mixed together for output. Sound latency varies with the sound card, but can be up to 500 milliseconds. To reduce latency:

- Play only one sound at a time (zero latency).
- Preload short sounds, if possible.
- Use uncompressed sounds (not SWA or IMA-compressed).
- Use one of the preferred PC sampling rates (11.025, 22.050, or 44.1 kHz) and not the variations that are Macintosh-specific (see Table 15-1).
- Use sounds of the same bit depth and sampling rate. The suggested sound format is 16-bit, 22.050 kHz, mono in most cases.
- Avoid changing the volume or performing sound fades when using multiple sounds.
- Combine the sounds in an external sound editor before importing into Director.
- Use a sound mixer with less latency, such as QT3Mix in D7, the DirectSound mixer (new in D7.0.1), the Beatnik Xtra, or MacroMix with RSX/DirectSound (in D6.x)

Sound output devices

Before we talk about audio sources, understand that there are two *mutually exclusive* sound output "devices" under Windows 95/98/NT: WaveOut and DirectSound ("device" refers to a virtual device driver, not the physical sound card). Only WaveOut is supported under Windows 3.1. DirectSound (part of the DirectX suite of Microsoft drivers) comes standard with later versions of Windows 98, and has been installed by many users on other Windows 95/98 systems. The latest version can be downloaded from *http://www.microsoft.com*. Windows NT supports the older DirectSound 3, which behaves as if the sound device is WaveOut.

All Windows audio must pass through one of these two devices, which sends the data onto the sound card driver and eventually the sound card itself. The sound output device can change dynamically at runtime, although only one device can be active at a given time. (Some IBM PCs support two simultaneous sound devices, but you can not rely on this unless all users have identical equipment.)

WaveOut supports only one input at a time. DirectSound 5 or later under Windows 95/98 can handle multiple simultaneous inputs. Windows NT with Service Pack 3 uses DirectSound 3, which, like WaveOut, supports a single input only.

Macromedia TechNote #13249, "Director sound playback mixing under Windows" contains detailed information (mainly regarding D6.5) on the differences between sound mixing under Windows 95/98 and Windows NT. It includes a helpful overview and informative diagrams of WaveOut and DirectSound mixing schemes:

> *http://www.macromedia.com/support/director/ts/documents/d6_sound_mixing01.htm*

Prior to Director 6, all Director sounds used WaveOut; although there *was* competition among multiple sources for the WaveOut device, there was not contention between the WaveOut and DirectSound devices (unless another application was using DirectSound).

Director 6 introduced support for RSX/DirectSound in addition to WaveOut. To reduce latency, D6.x Projectors keep the current device loaded even after a sound completes, unless specifically configured to release the device. If your sounds play during authoring but not in a Projector (and it is not caused by a missing Xtra) use the following to "offer-up" the device to WaveOut following playback via DirectSound, or vice versa:

```
set the soundKeepDevice = FALSE
```

Even if *the soundKeepDevice* is `FALSE`, the device is not released until all previously playing sounds complete. Set it to `TRUE` to decrease latency if only one output device is being used (see the following sections to determine whether WaveOut, DirectSound, or both are used by your Projector).

Potential conflicts from multiple sound input sources

Consider the following sources (classes) of audio used in Director for Windows and the sound mixing mechanism they use:

Director sounds

> Standard *#sound* and *#SWA* sprites in the Score, *puppetSounds*, and WAVE and AIFF sounds played via *sound playFile* commands are considered "Director sounds." These are always passed through Director's sound mixer (either MacroMix or QT3Mix) and may be played either via WaveOut or DirectSound as described under "Windows Sound Mixers." D7.0.1 can also use DirectSound via the new DirectSound mixer.

Flash sounds

> Sounds embedded in *#flash* members are always played via WaveOut and do not pass through Director's sound mixer.

VFW and QTW2 audio tracks

> Audio tracks contained in #digitalVideo members imported via `File ➤ Import` in D6.5 or earlier are known as "Traditional DV sounds." These are always played via WaveOut and do not pass through Director's sound mixer or the QuickTime for Windows Sound Manager.

QTW3 or later sound tracks

> Sound tracks contained in QTW3 #quickTimeMedia members imported via `File ➤ Import` (in D7) or inserted via `File ➤ Insert Media Element ➤`

`QuickTime 3` (in D6.5 or D7) are known as "QT3 sounds." These are always played via the QuickTime for Windows Sound Manager, which sends the audio to either DirectSound or WaveOut based on the *Sound Out* setting in the QuickTime Control Panel.

Third-party Xtras

Third-party Xtras may use their own sound mixer, Director's sound mixer, the QuickTime Sound Manager, or some combination of the three. (Beatnik uses MacroMix or its own sound mixer, but not the QTW Sound Manager.)

Playing multiple sounds of a single class is always supported, but may cause latency as they are mixed. Sound conflicts (where the second sound never plays) arise primarily from playing sounds of different classes under the following conditions:

- Playing sound from multiple sources to WaveOut under any Windows version. Director-based sounds in Windows 3.1, Flash-based sounds, and traditional DV audio (*#digitalVideo* sprites) can never be mixed with each other or with QTW3 (*#quickTimeMedia*) sprites.
- Playing sound from multiple sources to DirectSound 3 (which is the highest version supported under Windows NT).
- Playing sound to one device (WaveOut or DirectSound) when the other is actively playing a sound or "locked in" because *the soundKeepDevice* is `TRUE` (the default).
- Conflicts with the sound output of other applications. (See Macromedia TechNote #12180, "How does Director's use of sound on Windows affect other applications?")

There are several ways to avoid conflicts:

- Avoid playing sounds from a second source before sounds from the first source have completed. This allows Director to switch the sound device as needed, provided that *the soundKeepDevice* is FALSE.
- Use DirectSound 5 or later, which handles multiple input streams, as the output device for all sounds. In D6.x this requires that RSX and DirectSound 5 or later be installed and, if using QT3 sounds, that the QuickTime Control Panel specify DirectSound for *Sound Out*. This scheme will not work in D7.0, because RSX/DirectSound output is not supported for D7.0 sounds, nor will it work in D6.5 if the QuickTime Control Panel uses WaveOut, nor will it work under Windows NT with DirectSound 3 or under Windows 3.1. D7.0.1 with DirectSound 5 or later installed can use Macromedia's new DirectSound mixer without the need for RSX.
- Manually specify that Director should use QT3Mix, which sends sounds to the QuickTime Sound Manager where they can be mixed with QTW3 audio into a single stream before being sent to either the WaveOut or DirectSound device. (This requires a Windows 32 system with QTW3 installed and D6.5 or D7.)

Windows Sound Mixers

Sound mixers are a middle layer that mix multiple sound channels or input sources into a single data stream to be sent to the output device. The mixer for Director sounds is the only one that can be changed—QuickTime 3 sprites always play via the QuickTime Sound Manager, and other non-Director sounds always play directly to the WaveOut device (bypassing Director mixing).

Sound mixing with MacroMix

MacroMix transparently mixes multiple #sound, #SWA cast members, AIF and WAV files under Windows, but doesn't mix audio tracks from #flash, #digitalVideo, or #quickTimeMedia cast members.

MacroMix automatically configures itself based on the current sound card, although the settings in the *DIRECTOR.INI* file (see Table 15-11) can customize it. In most cases, the default *MixMaxFidelity* (99) is appropriate. The maximum number of mixable channels is determined by *MixMaxChannels* (the default is 4 in prior versions, and 8 in D7). See *Appendix D* in *Lingo in a Nutshell.* for details on working with the *DIRECTOR.INI* file.

Windows 3.1 Projectors always uses the 16-bit *MacroMix.DLL* to play sound. This DLL is bundled into Windows 3.1 Projectors (assuming that it is present when the Projector is built). It is unbundled temporarily into the Windows System folder at runtime and deleted when the Projector terminates.

MacroMix and other mixers for Windows 95/98/NT Projectors are implemented as Xtras in D7 and can be configured via Lingo at runtime. D6.5 and prior versions of Windows 95/NT Projectors used an internal version of MacroMix, which could be overridden using the *DLLname* option in the *DIRECTOR.INI* file in D6.5. (Except for the *DLLname*, all `[Sound]` settings in *DIRECTOR.INI* pertain only to MacroMix, and not to QT3Mix.)

In Director 4 through Director 6.0.2, MacroMix was the only sound mixer available under Windows. MacroMix is not a single mixer, it is a Sound API (application programmer's interface). The actual mixer used by MacroMix depends on the Director version and software configuration. D4, D5, 16-bit Projectors in D6, and D7 support only the WaveMix implementation of MacroMix. In D6.0 through D6.5, when using 32-bit Projectors, MacroMix uses its RSXMix implementation if RSX is installed.

WaveMix
: An implementation of MacroMix that uses WaveOut. It is a lowest common denominator mixer to ensure that multiple sounds can be mixed without requiring RSX, DirectSound, or QTW3, but it is characterized by latency and potential conflicts with other sound sources. To combat latency, refer to the tips under "Sound Mixing Latency." You may choose to initiate the sound early to make it play on time under Windows (in which case it would play early on the Macintosh). The 16-bit version of WaveMix is located in the *MacroMix.DLL* file. In D7, the 32-bit version is stored in *MacroMix.X32*.

RSXMix

An implementation of MacroMix that uses RSX, available only in Director 6.x. The system–level RSX service will use DirectSound if installed. It will use WaveOut if DirectSound is not installed or if *rsxDontUseDirectSound* is set to 1 in the *DIRECTOR.INI* file. RSX with DirectSound offered low latency but was tempermental, especially prior to D6.0.2, and not all users have RSX and DirectSound properly installed. RSXMix is not supported in D7.0 or later, but D7.0.1 includes a separate DirectSound mixer that does not require RSX.

QMix

A QT3-based mixer (typically referred to separately as QT3Mix) and described in the next section.

For more details on MacroMix, see Macromedia TechNote #13010, "How does Director play sound on Windows?" (pertains primarily to D6.5).

Sound mixing with QT3Mix

QT3Mix uses the QuickTime Sound Manager to mix Director sounds and requires that QTW3 (or the upcoming QTW4) be installed. QT3Mix is available only in D7 and D6.5 with the Service Pack installed, and is referred to as "QMix" in some Macromedia TechNotes.

The QuickTime Sound Manager will use DirectSound (if it is installed) under Windows 95/98; it uses WaveOut if DirectSound is missing or if running under Windows NT. The user can also set the preferred *Sound Out* device in the QuickTime Control Panel (and some developers report better results using WaveOut). There is no documented way to detect or switch the QuickTime Control Panel setting, but it is contained in the QuickTime Preferences file (*C:\Windows\System\QuickTime.qtp*) if you want to hack it.

QT3Mix allows Director and *#quickTimeMedia* sounds to play simultaneously under Windows 95/98/NT. QT3Mix is contained in the QT3Asset.X32 Xtra in D7, and the *QT3Mix.DLL* in D6.5 with the Service Pack. QT3Mix is the recommended (but not the default) mixer in D7.0. The D7 version of QT3Mix is much-improved over the D6.5 version, offering near-zero latency on faster computers (and reduced latency on slower Pentiums). See the following sections for complete details on activating QT3Mix under D6.5 and D7. In D7.0.1, if DirectSound 5 or higher is installed under Windows 95/98, the new DirectSound mixer should yield better performance than QT3Mix.

There is no need to change your Lingo when using QT3Mix instead of MacroMix. All sounds are played with the same familiar commands and methods.

QT3Mix is not supported under Windows 3.1 or with 16-bit Projectors under Windows 95/98/NT because they do not support QTW3. QT3Mix ignores the `[Sound]` settings in the *DIRECTOR.INI* file (except for the *DLLname* option).

RSX and DirectSound

RSX is a system-level service for Windows 32 systems from Intel. Obtain the latest version of RSX (*http://www.intel.com/ial/rsx/*) for best results when using D6.x. To determine whether RSX is installed, look for the *C:\Windows\System\RSX.DLL* file.

If RSX is enabled, a pair of red headphones appears in the Windows Start Menu tray.

DirectSound is a Microsoft sound driver that is part of the DirectX driver suite (which includes Direct3D, DirectDraw, etc.) and is not related to Director, per se. DirectSound version 5 or later is installed under Windows 98 by default, and most Windows 95 users have it too. DirectSound is compatible with Windows NT, but only up to DirectSound 3, which is implemented in software and has the same problems as WaveOut (latency and only one input source allowed). DirectSound is never supported under Windows 3.1.

D6.x required RSX to use DirectSound, and D7.0 never uses it. D7.0.1's new DirectSound mixer will use DirectSound without RSX.

Sound mixing with the Beatnik Xtra

The Beatnik Xtra (*http://www.headspace.com*) provides near-zero latency mixing with extremely low CPU overhead under both Macintosh and Windows, plus it includes sound effects, sound panning, support for additional sound formats (RMF, MIDI, and MOD in addition to AIF, WAVE, and AU) and much more.

The Beatnik Xtra is "Shockwave-safe" and is appealing for Shockwave delivery because it supports extremely compact sound formats for fast downloading and does not require RSX, QTW3, or DirectSound to be installed.

The major drawback is the licensing fee (which ranges from $495 to $1295 at press time, but may change) as you are not allowed to distribute the Lite version included with D7. Unlike QT3Mix, the Beatnik Xtra requires custom calls to play sounds, although the pro version includes premade Behaviors to play sounds.

Beatnik optionally uses the custom Headspace Audio Engine mixer (up to 32 channels without latency) or MacroMix (up to eight channels), but doesn't currently support QT3Mix or the QuickTime Sound Manager. Its MacroMix compatibility mode enables Beatnik audio to be intermixed with normal Director audio including SWA, *sound playFile* commands, and *puppetSounds*.

Sound Mixing Under Director for Windows

Table 15-5 summarizes the preferred sound mixer configurations under Windows for both Director and Shockwave if you are playing multiple sounds. It is a matter of considerable dispute whether the RSX/DirectSound combination available in D6.X is the preferred method of mixing sound. If RSX and DirectSound are installed properly, it works well, but some well-respected developers prefer using QT3Mix in D6.5, and using WaveOut by disabling RSX in D6.0.X.

Table 15-5: Preferred Sound Mixer Configurations

Environment	MacroMix and DirectSound	QT3Mix
D4, D5, SW4, SW5	WaveOut only[1,2,3]	N/A
D6.0.x, SW6.0	DirectSound[1] (requires RSX) or WaveOut[2,3]	N/A

Table 15-5: Preferred Sound Mixer Configurations (continued)

Environment	MacroMix and DirectSound	QT3Mix
SW6.0.1	WaveOut only (RSX ignores DirectSound)	N/A
D6.5 without Service Pack	Buggy, don't use.	Buggy, don't use.
D6.5 with Service Pack	DirectSound[1] (requires RSX) or WaveOut[2]	DLLname = QT3Mix.DLL[3]
D7.0, SW7.0	WaveOut only[4]	the soundDevice = "QT3Mix"[1,3]
D7.0.1, SW7.0.1	WaveOut (the soundDevice = "MacroMix") DirectSound[1] (the soundDevice = "DirectSound")	the soundDevice = "QT3Mix"[3]

[1] Preferred method in most Windows 95/98 configurations.
[2] Only supported method under Windows 3.1.
[3] Preferred method under Windows NT with SP3 and DirectSound 3.
[4] WaveOut is the only supported configuration in SW7.0 unless QTW3 is installed and the QT3Asset.X32 Xtra is downloaded.

MacroMix is the default mixer for Director-based sounds (*#sound*, SWA, AIFF, and WAVE files) in all versions of Director for Windows prior to D7.0.1. But its capabilities, whether it is the best method of mixing sounds, and how to override it varies with each version. Even if using a later version of Director, you'll find the following descriptions of sound mixing in earlier versions relevant.

Sound mixing in Director 4 and Director 5 for Windows

Director 4 and 5 always use MacroMix for Director-based audio. MacroMix will mix up to 8 Director sounds (the default is 4 in D4, D5, and D6.x), but exhibits marked latency in D4, D5, and D7. Traditional #digitalVideo sound tracks (QTW2 and AVI files) can not play simultaneously with Director sounds (AIFF, WAVE or sound cast members) in any version of Director for Windows, and are limited to one sound track per video.

In Director 4 and 5, *#digitalVideo* and Director sounds conflicted because both types use WaveOut. Whichever type took control of the WaveOut device first prevented sounds of the other type from playing. When one component finished playing its sounds, the other component could gain access to the WaveOut device.

Therefore, to allow a new type of audio to play, ensure that all sounds of the other type are stopped (see Table 15-4). Use the *puppetSound 0*, *sound close*, or *sound stop* commands to stop all Director sounds before attempting to play *#digitalVideo* sound tracks. Stop a video by setting *the movieRate of sprite* to 0, or disable its sound track using the *sound of member* property or *setTrackEnabled* command before attempting to play other Director sounds.

These limitations don't necessarily apply in D6, which can use DirectSound, or in D7 which doesn't typically use *#digitalVideo* members (*#quickTimeMedia* members are preferred).

Sound mixing in Director 6.0.x for Windows

Even in D6, Windows 3.1 Projectors always use the WaveMix implementation of MacroMix to play sounds to WaveOut, and therefore exhibit marked latency and conflicts with *#digitalVideo* audio (which uses WaveOut, as in D4 and D5).

For D6.x Windows 95/NT Projectors, the RSXMix implementation of MacroMix will be used if RSX is installed. RSXMix reduces latency substantially in the typical case where a button triggers a sound while a background track is playing. Absent RSX, the older WaveMix implementation (with the familiar latency and device conflicts) will be used.

Even RSXMix will resort to using WaveOut if DirectSound is not installed, but the RSX/DirectSound combination has the lowest latency (best performance). The performance is maximized by never releasing the DirectSound device (locking out *#digitalVideo* sounds, which always use WaveOut). To force RSX/DirectSound to release the sound device, you must set *the soundKeepDevice* to `FALSE`.

Some conflicts were reported between Director and RSX, especially prior to D6.0.2. If using D6.0 or D6.0.1, obtain the free update to D6.0.2 from Macromedia's site. RSX will use DirectSound by default, but because RSX and DirectSound have been plagued by installation and version issues, you can force RSX to use WaveOut (with reduced performance) by including the following line in the `[Sound]` section of your *DIRECTOR.INI* file:

```
[Sound]
rsxDontUseDirectSound = 1
```

Sound mixing in Director 6.5 for Windows

D6.5 was the first version to allow the developer to manually choose the mixer used for Director-based sounds. It allows you to specify QT3Mix using the *DIRECTOR.INI* file during Projector initialization.

If using D6.5, obtain the D6.5 Service Pack Update from: *http://www.macromedia.com/support/director/upndown/updates.html*.

The initial release of D6.5 (prior to the Service Pack) included an erroneous version of QT3Mix (a.k.a. "QMix"), mistakenly named "MacroMix.DLL." For Windows 3.1 Projectors, the bogus file overrode the default 16-bit *MacroMix.DLL* and prevented all sound from playing. Windows 95/NT Projectors ignored the external DLL and continued to use their internal version of MacroMix, but mistakenly set *the soundLevel* to 0.

The Director 6.5 Service Pack (unrelated to Windows 95/98/NT OS Service Packs) addresses the sound errors caused by the initial release of D6.5. It includes the same 16-bit *MacroMix.DLL* used in D6.0.2, and a copy of the correct 32-bit *QT3Mix.DLL* that is recognizable by Windows 95/NT Projectors.

QT3Mix allows Director and QT3 (*#quickTimeMedia*) sounds to be mixed together via the QuickTime Sound Manager regardless of the Windows 32 version or *Sound Out* setting in the QuickTime Control Panel. Unfortunately, it requires QTW3, still exhibits some latency, and can not mix *#digitalVideo* and *#flash* sounds. (See also Macromedia TechNote #13416, "Director 6.5 sound playback options, by cast member type.")

To use QT3Mix in D6.5+SP, include the following line in the `[Sound]` section of your *DIRECTOR.INI* file (or just remove the semicolon that acts to comment it out):

```
[Sound]
DLLname = QT3Mix.DLL
```

Include the *QT3Mix.DLL* file, and a copy of *DIRECTOR.INI* renamed to match your Projector's name, in the same folder as your Windows 95/98/NT Projector. If QTW3 isn't installed, no DLLname is specified, or *QT3Mix.DLL* does not accompany the Projector, QT3Mix will not load and MacroMix will be used instead.

The RSX/DirectSound combination has lower latency (better performance) than QT3Mix in D6.5. It allows QTW3 and Director sounds to play simultaneously under Windows 95/98 (but not Windows NT) if the QuickTime Control Panel specifies DirectSound as *Sound Out* (the default if it's installed). In this case, the two input sources are mixed by DirectSound instead of by the QuickTime Sound Manager.

Even if RSX and DirectSound are installed, setting the *rsxDontUseDirectSound* flag to 1 in the *DIRECTOR.INI* file, or configuring the QuickTime Control Panel to use WaveOut, would prevent Director sounds from mixing with QTW3 sound tracks if not using QT3Mix.

Sound mixing in Director 7.0 for Windows

Sound mixers are implemented as Xtras in Director 7, but should not be confused with the unrelated MIX Xtras used to import external media. At least one of the sound mixer Xtras must be included in the *Xtras* folder or bundled into the Projector in order to play sound in D7 under Windows. D7 allows sound mixers to be specified on the fly, whereas D6.5 configured the sound mixer during Projector start up only.

D7.0 includes two initial sound mixers for Windows: an implementation of MacroMix that always uses WaveOut, and QT3Mix (which uses DirectSound or WaveOut depending on the *SoundOut* setting in the QT3 Control Panel). D7.0.1 includes a third sound mixer, DirectSound, which offers improved mixing if DirectSound 5 or higher is installed. Unlike D6.x, D7.0.1 does not require RSX to access DirectSound; D7.0 and D7.0.1 ignore RSX in all cases.

MacroMix is contained in the *Xtras\Drivers\MacroMix.X32* Xtra. QT3Mix is contained in the *Xtras\QT3\QT3Asset.X32* Xtra and can be used to mix sounds even if you are not playing any QuickTime videos. The DirectSound mixer is contained in the *Xtras\Drivers\DirectSound.X32* Xtra included with D7.0.1.

There are two new properties related to sound mixing in D7—*the soundDevice* and *the soundDeviceList*—that affect the sound mixer selection, and thus indirectly affect whether the WaveOut or DirectSound device is used.

The *soundDeviceList* is a read-only list of the installed sound mixer Xtras:

```
put the soundDeviceList
-- ["MacroMix", "QT3Mix", "DirectSound"]
```

If no sound mixer Xtras are installed, *the soundDeviceList* returns an empty list. Although they may appear in *the soundDeviceList,* QT3Mix can not be used unless QTW3 is installed, and the DirectSound mixer cannot be used unless an appropriate version of the DirectSound drivers are installed. MacroMix, which doesn't depend on any system components, is always available if the *MacroMix.X32* Xtra is installed.

Use the *soundDevice* property to identify or set the current sound mixer. The default mixer depends on the installed Xtras and system components. In D7.0, before DirectSound mixing was offered, the default sound mixer was MacroMix, followed by QT3Mix if MacroMix was not installed. In D7.0.1, DirectSound is the default mixer if the DirectSound 5 drivers or higher are installed. If not, MacroMix becomes the default mixer, because older versions of the DirectSound drivers (such as DirectSound 3 under Windows NT) offer no benefit over MacroMix. If none of the necessary Xtras and system components are installed, *the soundDevice* will be 0, and sounds will not play.

When *the soundDevice* defaults to "DirectSound," it will offer the best available sound mixing. If *the soundDevice* defaults to "MacroMix" because DirectSound 5 is not installed, switching *the soundDevice* to "DirectSound" may kill sound playback if the DirectSound drivers are old or improperly installed.

If QTW3 is available, but DirectSound 5 or higher is not, setting *the soundDevice* to "QT3Mix" may be preferable to the default MacroMix mixer. QT3Mix provides reduced latency and conflict-free mixing of frame sounds, *puppetSounds*, *sound playFiles*, SWA, and QT3 sound tracks, regardless of whether RSX or DirectSound is installed. There may be a one-time delay of several seconds when changing *the soundDevice* to "QT3Mix." Use the following in D7.0.1 to take advantage of QT3Mix in the above scenario:

```
if the soundDevice = "MacroMix" and the quickTimePresent ¬
    and string (the soundDeviceList) contains "QT3Mix" then
  set the soundDevice = "QT3Mix"
end if
```

This example will work in both D7 and SW7 on all platforms. It will leave DirectSound as the default mixer if DirectSound 5 or later is installed, or attempt to load QT3Mix otherwise. It has no effect on the Macintosh. The checks in the example for *the quickTimePresent* and QT3Mix's presence are extraneous; setting *the soundDevice* to an unavailable mixer leaves its value unchanged (in some cases it may set *the soundDevice* back to its default). You can verify *the soundDevice* after attempting to set it.

The `Modify` ➤ `Movies` ➤ `Xtras` dialog box includes the MacroMix.X32 Xtra by default in D7. Although new movies created in D7.0.1 will also include Direct-

Sound.X32 by default, you may need to add it manually to the list of Xtras when upgrading movies from D7.0 to D7.0.1. Although the sound mixer Xtras are for Windows-only you should not remove them from the Xtras list, even on the Macintosh.

On the Macintosh, the only supported sound mixer is the MacSoundManager, which uses the Sound Manager system extension:

```
put the soundDevice
-- "MacSoundManager"
put the soundDeviceList
-- ["MacSoundManager"]
```

Sound mixing in Shockwave 6 and Shockwave 7

SW6.0 uses RSX with DirectSound to speed sound mixing, if available, and WaveOut otherwise. But very few users have both RSX and DirectSound properly installed, so sound latency and conflicts were common. SW6.0.1 uses RSX (if available) but always uses the WaveOut sound device, even if DirectSound is installed. SW6.x never uses QT3Mix and does not allow the developer to select the sound mixer manually.

SW7.0 and SW7.0.1 never use RSX. SW7.0 includes the MacroMix.X32 sound mixer in the default installation. In SW7.0, you can set *the soundDevice* to "QT3Mix" to reduce latency, provided that both the QT3Asset.X32 Xtra and QTW3 are installed. But the QT3Asset.X32 Xtra must be downloaded separately, and there is no convenient way to provide a QTW3 installer to Shockwave users (as there is when shipping a CD-ROM).

The DirectSound.X32 Xtra offers improved sound mixing in SW7 if DirectSound 5 or higher is installed (as it is on most Windows 95/98 systems, but not Windows NT). Set the *Download if Needed* checkbox under `Modify ➤ Movie ➤ Xtras` to auto-download the DirectSound.X32 Xtra for SW7.0 users (it is downloaded by default with SW7.0.1).

You can also use the Shockwave-safe Beatnik Xtra to mix sounds, as it doesn't require QTW3 or DirectSound, although it does have a licensing fee.

Controlling Sound Position and Playback

Director does not allow random access to any position within most sounds. SWA sounds can be paused and restarted, but AIFF, WAVE, and internal sounds always start from the beginning whenever played. They can be stopped, but not paused or cued. However, audio-only QuickTime or AVI movies allow full control over sound positioning and playback.

Cue points can be used for synchronization, but they are read-only. You can't jump to an arbitrary point in a AIFF or SWA file (although, see the wildly unsupported *setSoundTime* command in Table 15-14), but you can jump to an arbitrary point in a QuickTime or AVI movie. (MCI calls can be used to set an arbitrary location in a WAVE file under Windows, but it is not universally reliable.)

See Table 15-4 for commands to position sounds.

Sound Tools and Interface Options

Table 15-6 summarizes the interface options related to sounds.

Table 15-6: Sound-Related Interface Options

Action	**Command**
Edit or play the sound cast member in the external editor	File ➤ Preferences ➤ Editor (AIFF, MPEG3, snd, AU, SWA, WAVE) Edit ➤ Launch External Editor Command-, (Mac) or Ctrl-, (Windows)
View or edit a sound cast member's properties	Modify ➤ Cast Member ➤ Properties Double-click a sound cast member or sprite.
Import sound	File ➤ Import ➤ Sound (see Table 4-4)
Record a new sound (Mac only)	Insert ➤ Media Element ➤ Sound
Import SWA sound	Insert ➤ Media Element ➤ Shockwave Audio (retains SWA format) File ➤ Import ➤ Sound (D7; converts to non-SWA format)
Export Sound	Copy to Clipboard, or use Edit ➤ Launch External Editor, then save from your sound editor.
Export sound channels	Under File ➤ Export, use *Format: QuickTime Movie*, then choose *Options* and export Sound Channels 1 and 2.
Place a sound in the score	Modify ➤ Frame ➤ Sound Drag sound cast member to sound channel, or drag SWA or DV member to sprite channel.
Add cue points to a sound	See "Cue Points and Timing" and "Sound Editing Applications and Utilities" later in this chapter.
Create SWA cast member	See "Shockwave Audio (SWA)" later in this chapter.
Wait for sound or cue point	Modify ➤ Frame ➤ Tempo ➤ Wait for Cue Point. (In D5, use Wait for End of Sound or Wait for End of Digital Video options.)
Play sounds in the cast (internal, linked external, and SWA sounds)	*Play* button under Modify ➤ Cast Member ➤ Properties or Modify ➤ Frame ➤ Sound
Preview external sound files	*Play* button under File ➤ Import
Find sound cast members in Cast	Edit ➤ Find ➤ Cast Member ➤ Sound
Find SWA cast members in Cast	Edit ➤ Find ➤ Cast Member ➤ Xtra (D6) Edit ➤ Find ➤ Cast Member ➤ Shockwave Audio (D7)
Find Sound or SWA members in Score	Highlight member in Cast and use Edit ➤ Find ➤ Selection
Volume levels or mute a sound[1]	Control ➤ Volume, or volume button in Control Panel. Mute buttons to left of Score Sound channels.

[1] Cmd-Opt-M (Mac) or Ctrl-Alt-M (Windows) toggles *the soundEnabled* and does not affect *the soundLevel* property.

Sound Cast Member Properties Dialog Box

Some sound cast member's properties can be viewed and set via the *Sound Cast Member Properties* dialog box (see Figure 15-1) or set via Lingo, but a sound's sampling rate, bit depth, and number of channels are read-only in Director. The original sound must be modified in a separate sound editing program.

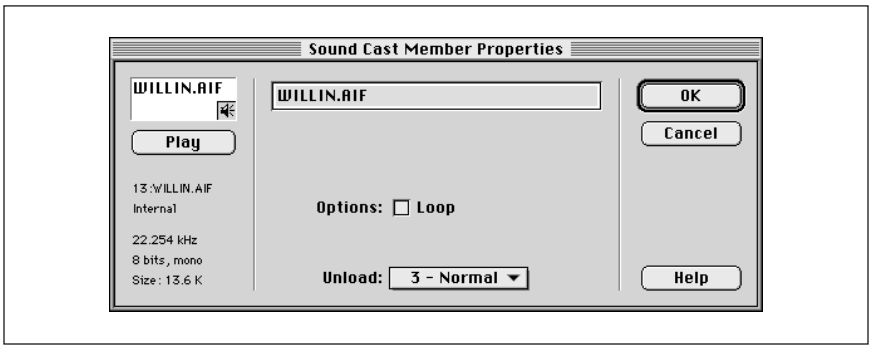


Figure 15-1: Sound Cast Member Properties dialog box

If multiple sound cast members are selected, the cast member properties dialog box will contain summary information, such as the total size of selected cast members.

The *Sound Cast Member Properties* dialog box can be used to play a sound. It also shows the sample rate (see *the sampleRate of member*), the number of channels (see *the channelCount of member*), and the bit depth (see *the sampleSize of member*). See Example 15-3 to determine the duration of an internal sound.

The size listed for internal sounds is accurate, but the size listed for externally linked files is merely the size of the cast member's header. See Example 4-6 to determine an external sound file's size.

The following sound properties are also shown in the dialog box:

Name and filename

If the sound is linked, an external filename is shown. Click on the name to browse to another filename. *The fileName of member* includes the complete path to the file and updates automatically for the current platform.

Loop

Loop controls whether Director obeys the loopback points set in an external sound editor. Absent any loopback points, the sound loops back to its beginning after it has completed. The Tempo Channel's *Wait for Cue Point:{End}* option causes Director to ignore the *loop* setting and play the sound until it ends. To wait indefinitely, use *Wait for Cue Point:{Next}* or the *soundBusy()* function. When creating a looping sound, ensure that the beginning and end of the loop combine seamlessly. External sounds won't loop automatically. See also the equivalent *loop of member*.

Unload

Set *Unload* to "Next" to keep a small sound in RAM (avoid this for larger sounds). Director often unloads sounds if it is low on memory, regardless of this setting. This setting has no effect on linked (streamed) audio.

Cue Points and Timing

Cue points are timing notations stored within sounds or digital video files. They were introduced in D6 and are used to synchronize audio or video with Score animations. Use sound tracks within QuickTime or Video for Windows when lip-synching or other close synchronization is required.

Director 6.5 and later supports cue points in WAVE files, as well as the AIFF, SWA, and digital video cue points supported in D6.0.x. Third-party Xtras such as the MPEG Xtra also support cue points.

Waiting for Godot's Audio

There are three ways to wait for audio:

- The Score's Tempo channel
- Checking the current playing time or audio state via appropriate Lingo properties, or using a Lingo function such as *soundBusy()*
- Waiting for a *cuePassed* event or checking the *isPastCuePoint()* function or *the mostRecentCuePoint* property

Tempo channel settings

The Tempo channel's *Wait for Cue Point* option can be used to wait for a sound to end or to reach a particular cue point. In the *Frame Properties: Tempo* dialog box (see Figure 15-2), choose the Sound channel or sprite channel to wait for, and choose from the list of available cue points within the sound, or *{Next}* or *{End}*.

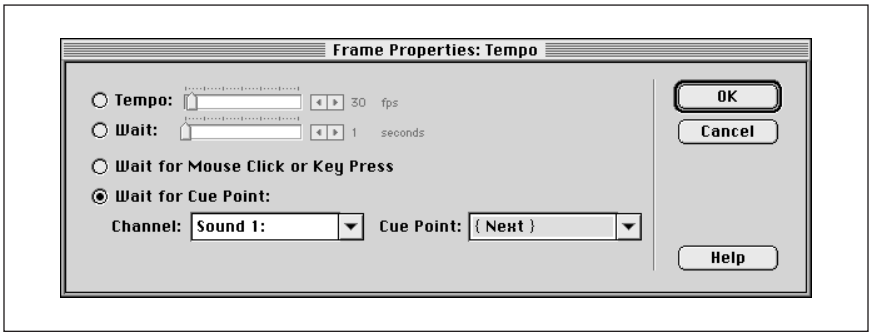


Figure 15-2: Tempo channel dialog box

Waiting for sounds via the Tempo channel in D5 (which used separate *Wait for End of Sound* and *Wait for End of Digital Video* options) locked out other events, such as mouse clicks. In D6, it locks out events for Custom Buttons, but otherwise

allows events to be processed. In D7, Custom Buttons are obsolete, and Director continues to process other events while waiting via the Tempo channel.

Waiting for sound via Lingo

Use Lingo for finer control over waiting for sounds. For example, the Tempo channel cannot be used to wait for sounds played via *sound playFile*. Use the *soundBusy( )* function instead, as shown in Example 15-4.

Example 15-4: Waiting for a Sound by Sound Channel Number

```
on exitFrame
  -- This waits for a sound in channel 2 to complete
  if soundBusy(2) then
    go the frame
  end if
end
```

As a general rule, you should *not* wait in a repeat loop, as it locks out all interactivity. Avoid this:

```
repeat while soundBusy(2)
  -- Waiting for the sound in channel 2 to end
end repeat
```

Avoid waiting for a sound to start unless you are sure it will actually start. For example, *sound playFile* will not give an error if you specify a missing or invalid filename; the sound will simply fail to play. The following will cause an infinite loop if the sound in channel 1 never starts.

```
puppetSound "someSound"
repeat while not soundBusy(1)
  -- Waiting for the sound to start
end repeat
```

To avoid an infinite loop in the prior example, trigger the *puppetSound* using an *updateStage* command before the *repeat* loop.

You can wait for a specific time in an SWA sprite by using *the currentTime* property in a script attached to the sprite of interest. Example 15-5 waits for 10 seconds (10,000 milliseconds) of the SWA to play. It will cause an infinite loop if *the currentTime* never reaches that point.

Example 15-5: Waiting for a Specific Point in an SWA Sound

```
on exitFrame me
  if the currentTime of sprite the spriteNum of me < 10000 then
    go the frame
  end if
end
```

Synchronizing with Cue Points

You can use cue points to wait in, leave, or jump to a particular frame in the Score. Create your cue points in a sound editing program with your Score layout and frame labels in mind. Create a cue point *before* the sound segment of interest

if you'll be jumping to a new frame when the cue point is reached. Create a cue point *after* the sound segment of interest if you'll be waiting in a frame until a cue point is reached.

To simplify your Lingo code, name your cue points the same as the frame label to which you wish to jump.

The Tempo channel's *Wait for Cue Point* option waits for a cue point in a Sound channel, digital video sprite, or SWA sprite. If the *Cue Point* option is {*Next*}, Director will wait for the next cue point to be reached. Use the {*End*} setting to wait for the end of a sound, even one without cue points. The {*End*} setting will not cause an infinite loop if a sound's *loop* option is set, but the {*Next*} setting will. The *Channel* parameter in the Tempo dialog box does not update automatically, so you must update it manually if you move a sound or sprite to a different channel.

Example 15-6 is a frame script that waits for a cue point. It then jumps to a frame whose label matches the cue point name.

Example 15-6: Using Cue Points to Synchronize with Score Animation

```
on exitFrame
  go the frame
end

on cuePassed me, channelID, cueNumber, cueName
  go frame cueName
end
```

Creating Cue Points

Cue points must be added to a sound before it is imported into Director. SoundEdit 16 (Macintosh only) can add cue points to AIFF and QuickTime files. You should use the SoundEdit v2.0.7 update at *http://www.macromedia.com/support/soundedit/updates*. Earlier versions may not create cue points correctly and will fail under Mac OS 8.

In D6.0.x, QuickTime cue point support was through a custom mechanism. QuickTime cue points appeared in SoundEdit as "markers" as they do for other file formats, but in MoviePlayer the same cue points appeared as a text track. (You must enable the track in MoviePlayer to see the text cue points, but Director reads the cue points even if the text track is disabled.)

Director 6.5 and 7 use the standard QT3 chapter tracks (which can be created and edited in any program that supports them) for cue point support in QuickTime.

AVI and WAVE cast members cannot contain cue points in D6.0.x, but Director 6.5 supports cue points in WAVE files (AVIs support cue points only if played via QT3). For example, cue points created in Sound Designer for Windows are

ignored by Director 6.0.x, but recognized by D6.5. WAVE files use markers ("MARK" chunks) to represent cue points. Any sound editor that supports markers can be used to create and edit cue points for D6.5.

To add cue points to a sound using SoundEdit 16 v2.0.7, follow these steps:

1. Open the sound file and click the location in the sound track at which you want to create the cue point.
2. Choose `Insert ➤ Cue Point`.
3. Enter a cue point name and/or change the cue point time. Director will always read cue points in milliseconds, regardless of the units used in SoundEdit.
4. You can move cue points by dragging them along the sound track, or delete them by dragging them off the sound track. Use `Windows ➤ Cue` to bring up SoundEdit's Cue Points inspector.
5. Save the file from SoundEdit in Audio IFF or QuickTime movie format or use the SoundEdit SWA Xtra from SoundEdit's `Xtras` menu to save an SWA file (requires PowerPC). Cue points added to a QuickTime file appear as text elements in a text track in QT2 and as chapter tracks in QT3.

To add cue points using Sound Forge or Cool Edit under Windows, see the *ReadMe Windows Sound Loop-Cue* file that comes with D6 and D7. To add cue points in Peak LE and for more tips on cue points, see *http://www.zeusprod.com/nutshell/cuepoints.html*.

To use cue points:

1. Import the asset into Director's Cast or `File ➤ Import` using `Insert ➤ Media Element` as appropriate.
2. Insert a sound into one of the Score's Sound channels or insert an SWA or QuickTime member into a sprite channel. *Sound playFile* and *puppetSound* also work with cue points.
3. Use the Tempo channel's *Wait for Cue Point* option to wait for a cue point or wait in a frame until a *cuePassed* event is sent to your *on cuePassed* handler.

Cue point caveats

Cue points trigger off the actual data buffers sent to and returned from the sound card under Windows. If there is no sound card (such as is common under Windows NT), cue points will not work.

A muted Sound channel in the Score will not load the specified sounds and those sounds won't generate cue point events. Similarly SWA or QuickTime sprites in muted sprite channels will not generate cue point events.

Cue point events *are* sent even if the volume is muted via the Control Panel or via Lingo.

Cue points near the end of a sound may not be recognized. Add silence to the end of the sound if necessary.

Avoid putting cue points beyond the end of an audio track in either a sound file or QuickTime movie. *The duration of member* reflects the position of the last cue point, not merely the end of audio data.

Cue Point Lingo

The cuePointNames of member property returns a Lingo list of cue point names for any cast member type that supports cue points. Unnamed cue points are shown as `"" (EMPTY)`. If no cue points are present, an empty list (`[]`) is returned. The *cuePointTimes of member* property returns a Lingo list of cue points times (in milliseconds), or an empty list (`[]`) if no cue points are present:

```
put the cuePointNames of member "mySound"
-- ["Intro", "Bridge", "Chorus", "", "Coda"]
put the cuePointTimes of member "mySound"
-- [4466, 7300, 13000, 17000, 21500]"
```

The *cuePointNames* and *cuePointTimes* of an SWA, sound, or QuickTime cast member are returned as empty lists (`[]`) until the sound is playing. See *the state of member* property.

The lists of cue point names and times is read-only, but can be manipulated with Lingo's list functions, such as:

```
set numCues = count (the cuePointNames of member "mySound")
set lastCue = getLast (the cuePointTimes of member "mySound")
```

The *isPastCuePoint()* function takes the general form:

```
isPastCuePoint (sprite n | sound n, cuePointNumber | cuePointName)
```

The first parameter is either a sprite channel or a sound channel. If the second parameter is a *cuePointNumber*, *isPastCuePoint()* returns a Boolean value indicating whether the current media playback position is beyond the specified cue point (regardless of how many times it may have passed that point). If the second parameter is a *cuePointName*, *isPastCuePoint()* returns an integer count of the number of times that the cue point with the given name has been passed (including multiple cue points with the same name).

The *mostRecentCuePoint* takes the form:

```
the mostRecentCuePoint of {sprite n | sound n}
```

It indicates the number of the last cue point passed for the specified sprite channel or sound channel. It returns 0 if no cue points have been passed.

Table 15-7 summarizes cue point-related operations.

Table 15-7: Cue Point Functions

Action	Command
Wait for cue point	Tempo channel, isPastCuePoint(), or see Examples 15-6 and 15-7
Determine names of cue points	the cuePointNames of member
Determine times of cue points	the cuePointTimes of member
Determine whether a cue point has been reached	isPastCuePoint()
Check the last cue point passed for a sprite or sound channel	the mostRecentCuePoint of sprite, the mostRecentCuePoint of sound
Trigger an event when a cue point is reached	*on cuePassed* event handler (see Examples 15-6 and 15-7)
Identify the sprite triggering a cue point	See the *the spriteNum of me* or *channelID* passed to the *on cuePassed* handler
Forces preloading of list of cue points (highly undocumented and unsupported in D6.5, and removed in D7)	forcePreloadCuePoints (member *whichMember*)

Cue Point Events

Director generates *cuePassed* events whenever it passes a media cue point in an appropriate sound or sprite. The beginning and end of the media do not automatically generate *cuePassed* events, although the Tempo channel's *Wait for Cue Point* option will wait for the end of a sound without cue points. As shown in Example 15-6, the declaration of an *on cuePassed* handler takes the form:

```
on cuePassed {me,} channelID, cuePointNumber, cuePointName
```

The *on cuePassed* handler receives three or four parameters as follows:

me

The script instance of the sprite that triggered the event, *me*, is sent to *on cuePassed* handlers in sprite scripts and frame scripts, but not to *onCuePassed* handlers in cast scripts or movie scripts. Use *the spriteNum of me* to determine the sprite's number.

channelID

channelID is the sprite channel or sound channel of the asset that reached a cue point. If the cue point is triggered by an SWA or QuickTime sprite, *channelID* is an integer from 1 to 120 in D6, or 1 to 1000 in D7. If triggered by a sound in a sound channel, channelID is *#sound1* or *#sound2* representing the Score's Sound channels; *#sound3* through *#sound8* represent *puppetSound* and *sound playFile* commands played in channels 3 through 8.

cuePointNumber

The number of the cue point within the asset triggering this event, starting at 1.

cuePointName

The name of the cue point triggering this event, or `EMPTY` (`""`) for unnamed cue points.

Example 15-7 can be used to analyze or diagnose cue point problems from any type of script. It automatically adjusts to whether 3 or 4 parameters are passed in (which depends on the script type). Ordinarily, you wouldn't use most of the information available, except perhaps the cue point name. See Example 15-6 for a typical cue point handler.

Example 15-7: Diagnostic Cue Point Event Handler

```
on cuePassed
  -- Sprite and Frame scripts receive four parameters
  -- Movie and Cast scripts receive three parameters
  if the paramCount = 4 then
    set me = param(1)
    put "on cuePassed handler reached for sprite" && ¬
      the spriteNum of me
  end if
  set channelID = param(the paramCount - 2)
  set cueNumber = param(the paramCount - 1)
  set cueName   = param(the paramCount)
  put "Channel ID:" && channelID
  put "Cue Number:" && cueNumber
  put "Cue Name:"   && cueName
  -- Print the cue point time
  case(channelID) of
    #sound1:
      set thisMember = the frameSound1
    #sound2:
      set thisMember = the frameSound2
    #sound3, #sound4, #sound5, #sound6, #sound7, #sound8:
      put "Cue point times not available for" && channelID
      set thisMember = 0
    otherwise:
      set thisMember = the member of sprite channelID
  end case
  if thisMember <> 0 then
    put "This cue time:" && getAt (the cuePointTimes of ¬
          thisMember, cueNumber)
  end if
end cuePassed
```

Shockwave Audio (SWA)

Shockwave audio (SWA) could be renamed *compressed audio*, because SWA can be used with a standalone Projector as well as with the Shockwave browser plug-in. For both Projectors and Internet delivery, SWA compression can be used to create external streaming SWA files or to compress *internal* sound cast members.

Compressing Sounds for SWA

When using SWA compression, you don't select a compression ratio—you select an output bandwidth. The throughput of users' Internet connections varies tremendously. You should pick a data rate that is sustainable over the slowest expected connection. Table 15-8 lists suggested output bit rates for SWA.

Even the highest quality SWA (160 Kbps) requires less than 20 KB/sec. Divide Kbps (1000 bits per second) by 8.192 (that is, 8×1024/1000) to convert to kilobytes per second.

Converting very large sound files to SWA may crash Director or SoundEdit. Refer to the Shocker-L archives (see the Preface) circa August 19, 1998 for comments about it.

Table 15-8: Shockwave Audio Delivery Rate Comparison

Delivery	Bit rate	Quality[1]
T1 or CD-ROM	64–160 Kbps (8–20 K/sec)	Equal to source material
ISDN	48–56 Kbps (6–7 K/sec)	FM stereo to CD-quality audio
28.8–56 Kbps modem[2]	16–32 Kbps (2–4 K/sec)	FM mono or good quality AM
14.4 Kbps modem[2]	8 Kbps (1 K/sec)	Telephone

[1] All SWA is decompressed as 16-bit audio. Stereo sounds are automatically folded (flattened) to monaural if an output rate of 32 Kbps or lower is used.
[2] Only external SWA files can be compressed to 8, 16, or 24 Kbps as is necessary for streaming over a modem. Internal Director sounds can be compressed only to 32 Kbps or higher.

If you expect approximately 2 K/sec through a 28.8 Kbps modem, you should use a compression rate of 16 Kbps (equal to 2 K/sec). The goal is gapless delivery, but if the stream cannot keep up, the audio will pause or drop out.

Internal sound cast members can be compressed to rates of 32 to 160 Kbps (4 to 20 K/sec). External SWA sounds can be compressed as low as 8 Kbps (1 K/sec). You might use 8 Kbps or 16 Kbps SWA for uninterrupted streaming of large sounds over a 28.8 Kbps modem and 64 Kbps for high-quality smaller internal sounds.

During streaming or downloading, only the bandwidth (i.e., *the bitRate of member*) is important. Streaming sound is discarded as it is played, so it doesn't use much RAM.

The disk size of an SWA file is only relevant if calculating the CD-ROM space required when using local SWA files. Its size depends only on the duration of the source audio and the bit rate chosen for SWA compression, regardless of the source material's sample rate and sample size. The size of an SWA file on disk can be calculated in KB as:

```
(the duration of member) * (the bitRate of member) / 8192
```

Compressing internal sounds

Standard internal sound cast members can be compressed as follows:

1. Fully import (unlinked) sounds into Director using `File` ➤ `Import` ➤ *Standard Import*.
2. Enable compression under `Xtras` ➤ `Shockwave for Audio Settings` and choose 32, 48, 56, 64, 80, 112, 128, or 160 Kbps as the final output rate.

3. Compression does not occur until the DCR or CCT file or Projector is created using `File ➤ Save As Shockwave Movie`, `Xtras ➤ Update Movies`, `File ➤ Create Projector ➤` *Options* `➤` *Compress (Shockwave format).*
4. If Shockwave compression is not enabled, internal sounds in DCR and CCT files are compressed about 30% using LZW compression. Internal sounds in DIR, DXR, CST, and CXT files are never compressed.

Internal non-streaming sounds that are SWA-compressed using `Xtras ➤ Shockwave for Audio Settings` are blown up fully into RAM when needed. They use the standard properties for *#sound* cast members, not those for *#SWA* members, as shown in Tables 15-4 and 4-10. Likewise in D7, SWA cast members imported via `File ➤ Import` are converted to Director's internal sound format. Insert SWA via `Insert ➤ Media Element ➤ Shockwave Audio` to retain the SWA format.

Compressing external sounds

External SWA sounds can be compressed to 8, 16, or 24 Kbps (which are designed for very low bandwidth), in addition to the higher quality rates (32–160 Kbps) available for internal sounds.

Director for Macintosh compresses internal sounds only. Use SoundEdit 16 or Peak LE to compress external sounds on the Macintosh.

The steps for SWA compression of external sounds in SoundEdit 16 (PowerMac-only) are:

1. Import or create sound files within SoundEdit v2.07.
2. Within SoundEdit, choose `Xtras ➤ Shockwave for Audio settings`.
3. Compression occurs when using `File ➤ Export` (sound format *Shockwave Audio*).
4. Import sounds as linked SWA cast members into Director using `Insert ➤ Media Element ➤ Shockwave Audio`.

For an SWA Tutorial, examples, and SWA players, click the "Working With Shockwave" option and then the "Download the example movies" option at *http://www.macromedia.com/support/soundedit/*. The following two links include the SWA Xtras for SoundEdit 16 (also included with the D6 Studio).

SWAtomator—batch processes files to SWA:

http://www.macromedia.com/support/soundedit/SE16SWA.hqx

SoundEdit 16 updater to version 2.0.7 (supports Mac OS 8.x):

http://www.macromedia.com/support/soundedit/SE16v207.hqx

To export SWA from Peak LE:

1. Install Peak LE from the *Peak LE 2.0* folder included on the Director 7 Shockwave Internet Studio CD.
2. Copy the SWA Export Xtra from that same folder to the *Peak LE Plugins* folder (where Peak LE is installed).
3. Choose *Shockwave .swa* format under `File ➤ Save As`.

SWA compression of external sounds in Director (Windows only) requires a Pentium and operates on WAVE files. Choose `Xtras ➤ Shockwave for Audio settings` and then `Xtras ➤ Convert WAV to SWA`.

SWA compression hints

SWA uses MPEG3 compression (SWA files can be previewed in MacAmp or WinAmp) and is optimized to deliver high quality audio at reasonable bandwidths, but it also delivers fair quality audio at minuscule bandwidths. Whether compressing internal or external sounds, use at least 16-bit, 22.050 kHz source audio. Do *not* downsample the audio to 8-bit first. After SWA compression, 8-bit, 11 kHz monaural audio occupies the same space as 16-bit, 22 kHz monaural audio (or even 16-bit, 44 kHz).

There is no benefit to reducing either the bit depth or the sample rate before compression. Quite the contrary, higher fidelity source audio results in higher quality SWA without any additional bandwidth. All SWA-compressed sounds *reconstruct* (decompress) into 16-bit audio regardless of the source material's bit depth (which is why *the bitsPerSample of member* property always returns 16).

The numChannels of member is 1 (monaural) if *the bitRate of member* is 32000 bps or less. SWA properties can be checked only after the SWA begins playing, as indicated by *the state of member* property (see Table 15-9).

Use the highest output rate that your Internet connection will tolerate (see Table 15-8). Output rates below 32 Kbps are intended only for compatibility with slower modems. As compression increases, the absolute savings are only marginally better.

For example, a CD-quality (16-bit, 44 kHz, stereo) source file requires 176 K/sec (1400 Kbps). Compressing it to 160 Kbps reduces the size by 89% with no detectable loss of quality. Compressing it to 32 Kbps (4 K/sec) saves a whopping 172 K/sec and it can still play over a fast modem. But compressing it to 8 Kbps (1 K/sec) saves only an additional 3 K/sec, and the loss in quality to achieve the marginal savings is *immense* (according to Macromedia).

SWA Decompression

SWA decompression is processor-intensive, so SWA is most appropriate over the Internet, where download time is at a premium. For local content, SWA may simply hinder performance, especially on slower machines.

Decompression time is not significantly different at different data rates (as with most compression schemes, decompression is much faster than compression). The propensity to drop sounds seems independent of the bit rate. On 68K Macs (which require an FPI to play SWA), SWA decompression ignores some of the data when using 8 Kbps and 16 Kbps bit rates. This provides adequate performance at the expense of lesser quality than on PowerMacs.

All SWA is decompressed as 16-bit audio (which is why you shouldn't downsample to 8-bit). Internal sounds that were compressed with SWA are reconstructed in their entirety in RAM before playback; they are blown up to their original size, so a 44 kHz source will occupy twice the memory of a comparable

22 kHz source once decompressed. Regardless, streaming external sounds don't remain in memory—they use only a small temporary buffer—so the size is only relevant for internal sounds.

SWA Lingo

Table 15-9 lists the commands and properties pertaining to SWA cast members. All the properties are read-only and apply to externally linked *#SWA* cast members, not to internal cast members compressed when the Director movie or castLib is compressed as a whole. All SWA commands are new as of D6; although some were supported in Shockwave for D5, none were supported during authoring in D5.

Table 15-9: Shockwave Audio Lingo

Property	**Usage**
the bitRate of member[1]	0 (not ready) | 8000 | 16000 | 24000 | 32000 | 48000 | 56000 | 64000 | 80000 | 112000 | 128000 | 160000
the bitsPerSample of member[1,2]	Bit depth of expanded media, *not* the bit depth of the original file that has been SWA encoded. It always returns 0 (not ready) or 16.
the copyrightInfo of member[1,2]	Copyright text for sound file.
the cuePointNames of member[1,2]	List of names of cue points.
the cuePointTimes of member[1,2]	List of cue point times in milliseconds.
the currentTime of sprite[1,2]	Current point in playback in milliseconds.
the duration of member[1]	Duration of SWA file in seconds (different units than property of the same name for #digitalVideo, #quickTimeMedia, and #transition members).
getError (member *swaMember*)	Error status for SWA cast members. The integer value returned by *getError( )* corresponds to the string returned by *getErrorString( )*: 0: returns EMPTY string ""[3] 1: "memory" 2: "network" (or "Network software error") 3: "playback device" 99: "other"
getErrorString (member *swaMember*)[3]	See *getError( )*.
isPastCuePoint(member, cueID)[2]	Returns a positive integer if the cue point has been passed.
the mediaReady of member	Indicates whether media has been completely downloaded. Appropriate only for internal nonstreaming SWA-compressed sounds.
the mostRecentCuePoint of member[2]	Returns number of most recent cue point passed.
the numChannels of member[1,2]	Number of channels (usually 1 or 2). Returns 1 when *the bitRate of member* is <= 32000.

Table 15-9: Shockwave Audio Lingo (continued)

Property	Usage
pause (member)	Pauses SWA stream, but not instantly.
the percentPlayed of member	Percentage of bytes played (0 to 100). Should be less than or equal to *percentStreamed*.
the percentStreamed of member	Percentage of bytes streamed from server (0 to 100).
play (member)	Begins playing SWA (initiates preload, too).
preLoadBuffer (member *swaMember*)	Begins preloading amount of data specified by *the preLoadTime of member*. Use *stop()* to "rewind" the media after preloading before using *play()*.
the preLoadTime of member	Duration of SWA audio (not download time) in seconds to be downloaded before playback begins (prevents skipping).
the sampleRate of member[1,2]	Sample rate of original sound source (in Hz) before compression, such as 11025, 22050, or 44100.
the soundChannel of member[1,2]	System sound channel in which to play SWA (0 uses highest available channel). Avoid using 1 and 2, the Score Sound channels.
the state of member	*The state of member* must be 2, 3, 4, or 5 before checking other SWA properties accurately: 0: Stopped 1: Preloading 2: Preloading completed 3: Playing 4: Paused 5: Done 9: Error 10: Insufficient CPU
stop (member)	Stops SWA stream, but not instantly. Also rewinds SWA.
the streamName of member[1,2]	URL of SWA file. Can be local. (Same as *URL of member*.)
the url of member[1]	URL of SWA file. Can be local. (Same as *streamName of member*.)
the volume of member	Volume of SWA, from 0 to 255.

[1] Returns 0, [], EMPTY, or meaningless data unless *the state of member* is 2, 3, 4, or 5.
[2] Appears for the fist time in Director 6. Not supported in Shockwave for Director 5.
[3] *GetErrorString(0)* returns the EMPTY string in my tests, not "OK" as claimed by Macromedia's documentation.

Pausing an SWA and playback gaps

When you use *pause()* or *stop()* to halt an SWA, a certain amount of audio data is still in the SWA buffer and continues to "play out." Therefore, the sound does not pause immediately. To stop the sound from being heard, you must set *the volume*

of member to 0. When you use *play()* to start the sound again (don't forget to reset the volume), the sound will have skipped the portion that was played out of the buffer after the last *pause()* command. No ideal solution exists.

If an SWA sprite is streaming, there may be times when the Internet connection does not provide data fast enough and gaps in the audio occur. Even though the audio is momentarily interrupted, *the state of member* still returns 3 (playing).

Likewise, the *soundBusy()* function indicates whether a sound channel has been allocated and is presently "in use," not whether sound is currently audible.

There is no random access to an arbitrary point in an SWA stream, which is incompatible with streaming playback. The data would have to be either stored to disk or preloaded to allow such access, as is done by various MP3 players. However, see the highly unsupported Din Xtra commands in Table 15-14.

Streaming live audio sources via Shockwave

According to John "jd" Dowdell of Macromedia, there are two alternatives to encode and broadcast SWA in realtime:

- Use the Telos AudioActive (*http://www.audioactive.com*) real-time encoding and multicasting equipment. It works under Windows and with the SW6.0 on the Mac. (It doesn't work with SW6.0.1 and Netscape 3 and 4 on the Macintosh; the audio stream downloads but never plays.)
- Use RealAudio's real-time compression and multicasting hardware. The RealAudio Xtra requires installation in the browser plug-in folder. You can instead use a RealPlayer element in a web page and use LiveConnect and ActiveX scripting from Shockwave to control it. (Some have reported poor results with this solution.)

Details on these solutions using Shockwave 7 were not available at press time. The RealAudio Xtra won't work in SW7 unless a Shockwave-safe version is made available by Real Networks, and no such plans appear likely.

Other Sound-Related Lingo

Table 15-10 lists system-level sound-related Lingo properties and commands.

Table 15-10: System-Level Sound-Related Lingo

Command	Usage
beep {*n*}	Beeps *n* times (*alert* also causes a beep). If *the soundEnabled* is FALSE, beep flashes the Macintosh menubar. *n* defaults to 1, and multiple beeps don't usually work under Windows.
the beepOn	If TRUE, Director beeps when the user clicks on an inactive sprite (one without a mouse script attached). Useful for debugging.
sound close *channel*	Closes the specified sound *channel*. This requires the sound buffer to be reallocated for the next sound.

Table 15-10: System-Level Sound-Related Lingo (continued)

Command	Usage
sound stop	Stops the sound playing in the specified *channel*.
the soundEnabled	If FALSE, all sounds are muted.
the soundLevel	System volume from 0 (mute) to 7 (loudest).
the multiSound	If TRUE, the computer supports stereo sound.

The *DIRECTOR.INI* file has numerous settings that affect the audio buffers and sound-mixing under Windows. Ordinarily, you shouldn't change any except the *DLLname* and *rsxDontUseDirectSound*, which are both obsolete in D7, anyway. Refer to Appendix D in *Lingo in a Nutshell* for additional details on setting the items listed in Table 15-11.

Table 15-11: DIRECTOR.INI File Sound-Related Settings

Command	Usage
DLLname (D6.5 only)	Default is MacroMix.DLL. Change to QT3Mix.DLL to use QT3 Sound Mixing (requires D6.5 with Service Pack).
rsxDontUseDirect-Sound (D6.x only)	If set to 1, RSX will output to WaveOut instead of DirectSound. Default is 0. SW6.0 treats this as 0; SW6.0.1 treats it as 1.
MixMaxChannels	Maximum number of sound channels allowed to be mixed. Default is 8 in D7 and 4 in previous versions; max is 8.
MixMaxFidelity	Maximum sound fidelity that MacroMix will need to mix. The default, 99, determines max on the fly: 0: 22.050 kHz, 8-bit, mono 1: 22.050 kHz, 8-bit, stereo 2: 44.100 kHz, 16-bit, stereo 99: Switch formats on the fly (default)
MixServiceMode	0: Interrupt Mixer based on MixIntPeriodMs and MixIntResolutionMs (default) 1: Use polling to drive mixer (may cause drop out if other tasks hog CPU access) 2: Use waveout buffer-completion callback to drive mixer. Set MixBufferBytes to multiple of 1024
MixWaveDevice	Ranges from 0 (default) to the number of devices – 1.
MixBufferBytes	Defaults to 0. Set MixBufferMs to 0 and then set MixBufferBytes to multiple of 1024 to specify a buffer size in bytes.
MixBufferCount	Number of mixing buffers from 2 to 16 (default is 4).

Table 15-11: DIRECTOR.INI File Sound-Related Settings (continued)

Command	Usage
DLLname (D6.5 only)	Default is MacroMix.DLL. Change to QT3Mix.DLL to use QT3 Sound Mixing (requires D6.5 with Service Pack).
rsxDontUseDirect-Sound (D6.x only)	If set to 1, RSX will output to WaveOut instead of DirectSound. Default is 0. SW6.0 treats this as 0; SW6.0.1 treats it as 1.
MixBufferMs	Defines sound buffer length in milliseconds. Default is 200 milliseconds (size in bytes varies with sound format). To define buffer size in bytes using MixBufferBytes, set MixBufferMs to 0.
MixIntPeriodMs	Interrupt interval, defaults to 200 milliseconds. Used only when MixServiceMode is 0.
MixIntResolutionMs	Interrupt duration, defaults to 50 milliseconds. Used only when MixServiceMode is 0.
SoundLevel0 through SoundLevel7	Specifies "wave output" volume corresponding to *the soundLevel* property. Default values for the soundLevel = 0 to 7 are shown:[1] SoundLevel0 = 0 SoundLevel1 = 24770 SoundLevel2 = 35030 SoundLevel3 = 42903 SoundLevel4 = 49540 SoundLevel5 = 55388 SoundLevel6 = 60674 SoundLevel7 = 65535
HighSpoolBufferMs	Length of one 16-bit spool buffer; default is 1500 milliseconds.
LowSpoolBufferMs	Length of one 8-bit spool buffer; default is 2500 milliseconds.
SpoolBufferAlloc	0: Allocates/deallocates spool buffers dynamically, when sound starts/stops (default) 1: Allocates spool buffer once at startup and keeps them it entire session
SpoolBufferCount	Number of spool buffers to allocate, from 2 (default) to 10.

[1] *The soundLevel* settings in the *DIRECTOR.INI* file can range from 0 to 65,535, but the output response is nonlinear. The default settings for *soundLevel1* through *soundLevel3* are inaudible on some PCs.

Volume Levels and Sound Fades

Volume levels vary tremendously across platforms and even on different machines running the same OS. Perform tests to determine an appropriate volume level and then record all your sounds for a project at the same level. You can later adjust their relative volumes in Director (this is simplified greatly if all background sounds or voiceovers are played in a specific channel).

There are many commands that control sound volume in Director, including system-level volume controls, plus those at the sprite or cast member level. On the Macintosh, Lingo can set the master volume, but under Windows, Lingo controls only the "wave output" volume.

Provide a volume control with a mute option to the user. Macromedia provides an example volume slider in the D6 Behavior Library. You can use the keyboard characters 0 through 7 to set *the soundLevel*, as shown in Example 15-8.

Example 15-8: Setting the soundLevel via the Keyboard

```
on keyDown
  -- 0 though 7 set the soundLevel.  Keys 8 and 9 set it to 7.
  if charToNum(the key) >= 48 and charToNum(the key) <= 58 then
    set the soundLevel = min(integer(the key), 7)
  end if
end keyDown
```

It is exceedingly rude to increase the system volume level automatically. If necessary, check *the soundLevel* or sprite volume via Lingo and suggest that the user change it. If you do set *the soundLevel*, set it to 5, not 7.

Table 15-12 lists commands that pertain to sound volumes in Director. See also `Control ➤ Volume` and the volume button in the Control Panel.

Table 15-12: Volume-Related Lingo

Command	**Usage**
sound fadeIn *channel* {, *ticks*}	Fades volume up from 0 to the current *volume of sound* setting, not up to 255, as in D5. See the D6 *ReadMe*.
sound fadeOut *channel* {, *ticks*}	Fades volume down from the current *volume of sound* setting to 0, not from 255 to 0, as in D5. See the D6 *ReadMe*.
the sound of member *videoMember*	Boolean indicating whether sound is enabled for a digital video or Flash cast member.
the soundEnabled	If FALSE, all sounds are muted.
the soundLevel	System volume from 0 (mute) to 7 (loudest). See Table 15-11.
the visibility of sprite	Muting a Sound channel or sprite channel in the Score prevents the sound, SWA, or digital video sprite from playing.
the volume of member	Volume level for SWA members (0 to 255). Doesn't work for Flash.
the volume of sound	Volume level for Sound channel 1 or 2 (0 to 255).
the volume of sprite	Volume level for digital video, AVI, QT2 in D6, QT3 in D7, and SWA sprites. Nominal range is 0 to 255, but can be set much higher.
the volumeLevel of sprite	Volume level for QT3 digital video sprites in D6.5. Use *volume of sprite* in D7. Nominal range is 0 to 255, but can be set much higher.

Volume levels

The soundLevel command sets the volume for the overall system on the Macintosh and it matches the settings in *Sound* or *Monitors & Sound* Control Panel. Under Windows, *the soundLevel* controls the "wave output" volume matching the

SoundLevel0 through *SoundLevel7* settings in the *DIRECTOR.INI* file (see Table 15-11). Note that human audio perception is nonlinear. The default settings for *SoundLevel0* through *SoundLevel3* are usually indistinguishable from each other and undesirably soft. Consider respecifying the range of *SoundLevel1* to *SoundLevel7* from 45,000 to 65,535 in the *DIRECTOR.INI* file for more useful control via Lingo's *the soundLevel* property.

For complete control of the master volume under Windows, either open the Windows Sound mixer or use a third-party Xtra.

Burak Kalayci's bkMixer Xtra adjusts the Windows master volume level:

http://www.updatestage.com/xtras/bkmixer.html

Buddy API Xtra—see the *baGetVolume()* and *baSetVolume()* methods:

http://www.mods.com.au/budapi/

Some Windows sound cards' volume mixers can be controlled via *mci* commands. Bear in mind that the volume can also be changed by the user on some speakers with external volume knobs.

The soundLevel affects all sounds; you can adjust the relative volumes of individual members or sprites using the commands shown in Table 15-12.

There is no volume control over Flash sprites beyond shutting sound off using the *sound of member* property. Use native Director sounds instead.

Modal MIAWs reportedly prevent *the soundlevel* from being set under Windows NT in D7.0.

Sound fades

Prior to D6, *sound fadeIn* and *sound fadeOut* always faded between the minimum and maximum volumes (0 to 255). In D6 and later they fade from the current *volume of sound* for the specified channel towards either 0 or 255 as appropriate.

Interrupting sound fades under Windows tends to freeze *the volume of sound* at the level it held when the fade was interrupted. For example, if a sound terminates in the Score before a *sound fadeOut* completes, it might lock the volume for that channel to a near-zero level. Use this to reset the problem:

```
set the volume of sound channel = 255
```

You can manually construct your own *fadeIn* and *fadeOut* commands to fade between two arbitrary volume levels and avoid the buggy sound fade commands altogether. See *http://www.zeusprod.com/nutshell/fade.html*.

Some conflicts have been reported when fading sounds with different sampling rates at the same time. Always use sounds of the same rate at the same time.

Sound-Related Xtras

There are two broad categories of sound Xtras: those required by Director to play external sounds and those that add additional sound-related features of interest, but are not mandatory for most users.

Xtras needed to play external sounds in Director 6 and 7

Director 5 does not require any Xtras for sound playback. In Director 6 and 7, as with most external media, MIX Xtras are required to access external sounds at runtime. Note that in D6 the MIX Services and Sound Import Export Xtras are automatically added under `Modify` ➤ `Movie` ➤ `Xtras` when your cast includes linked sounds, but sometimes additional Xtras are needed. These Xtras, plus the SWA Xtras, are added by default to all D7 Projectors unless deleted from the `Modify` ➤ `Movies` ➤ `Xtras` list. This list also includes the MacroMix and Direct-Sound mixers. Regardless, I recommend against bundling Xtras with your Projector, so you should distribute the Xtras listed in this section with your Projector in a separate *Xtras* folder.

If all your sounds are embedded (unlinked) internal sounds, you don't need any Xtras unless you are using internal SWA compression.

Following is a list of Xtras needed in certain situations. See Table 15-13 for the exact names of the Xtras needed on the various platforms. The Xtras can be found in the *MIX*, *Media Support*, *Net Support*, *Device,* and *QT3* subfolders within the *Xtras* folder where Director is installed.

Playing any sounds in D7 or SW7 under Windows
: D7 and SW7 for Windows require the MacroMix.X32, DirectSound.X32, and/or QT3Asset.X32 Xtras, depending on the user's installed software.

Linked sounds played via sound playFile, puppetSound, and the Score
: To use *sound playFile* or to play any externally linked sound cast members via the Score Sound channels or via *puppetSound*, include the MIX Services and Sound Import Export Xtras with your Projector.

Local SWA audio played from a Projector
: To play SWA from a local drive, include the MIX Services, Sound Import Export, SWA Streaming, SWA Decompression, and NetFile Xtras with your Projector. (NetFile is needed even when not using the Internet.)

SWA streaming over the Internet played from a Projector
: To play SWA from a remote server via a Projector, include the MIX Services, Sound Import Export, SWA Streaming, SWA Decompression, NetFile, INetURL, and NetLingo Xtras (and the NetManage Winsock Lib, for PowerPC only) with your Projector.

Table 15-13 lists sound-related Xtras.

Table 15-13: Xtras Needed for Sound Playback.

PowerPC	**Mac 68K**	**Win 32**	**Win 16**
MIX Services	MIX Services	mix32.X32	mix16.X16
Sound Import Export	Sound Import Export 68k	Sound Import Export.X32	mixsound.X16
SWA Streaming PPC Xtra	SWA Streaming 68K Xtra	swastrm.X32	swastrm.X16

Table 15-13: Xtras Needed for Sound Playback. (continued)

PowerPC	Mac 68K	Win 32	Win 16
SWA Decompression PPC Xtra	SWA Decompression 68K Xtra	swadcmpr.X32	swadcmpr.X16
NetFile PPC Xtra	NetFile 68K Xtra	NetFile.X32	NetFile.X16
InetUrl PPC Xtra	None needed	INetUrl.X32	INetUrl.X16
NetLingo PPC Xtra	NetLingo 68K Xtra	NetLingo.X32	NetLingo.X16
NetManage WinSock Lib	None needed	None needed	None needed
MPEG 3 Import Export	MPEG 3 Import Export 68K	MPEG3 Import Export.X32	N/A
N/A	N/A	QT3Asset.X32[1] (contains QT3Mix)	N/A
N/A	N/A	MacroMix.X32[1]	N/A
N/A	N/A	DirectSound.X32[2]	N/A

[1] D7 only. MacroMix.X32 is bundled with SW7.0.
[2] D7.0.1 only. MacroMix.X32 and DirectSound.X32b are bundled with SW7.0.1.

If you choose to bundle Xtras with your Projector instead of shipping them separately, you can check the *Check Movies for Xtras* option in the D6 Projector creation dialog box in all of the previous cases. If playing SWA files either locally or remotely, you can also check the *Include Network Xtras* D6 Projector option. Instead of using these checkboxes, you can manually add the specified Xtras to your Projector file build list. In D7, individual Xtras can be flagged for inclusion under `Modify` ➤ `Movie` ➤ `Xtras`. All the previously listed Xtras are included by default for all D7.0.1 movies, whether needed or not. (Again, I recommend removing them from the list and placing them in an *Xtras* folder.) D7.0 movies will not include the DirectSound.X32 Xtra until upgraded to D7.0.1.

Xtras needed for sound in Shockwave

The equivalent of the MIX Services, Sound Import Export, and network-related Xtras (NetFile, INetURL, NetLingo, and the NetManage Winsock Lib for PowerPC) are built into Shockwave 6 and 7. These Xtras are needed only during authoring or in a Projector.

To play SWA from within Shockwave, the SWA Streaming and SWA Decompression Xtras must be installed in the Shockwave *Xtras* folder. These Xtras are both installed by default with Shockwave 6 and 7. Shockwave 7 for Windows also installs MacroMix.X32. SW7.0.1 installs DirectSound.X32 in addition. These Xtras are needed to mix sounds under Windows.

See Chapters 10 and 11 for more details on Xtras and Shockwave.

Other third-party sound-related Xtras

Besides the Xtras mentioned throughout this chapter, here are some that support more esoteric functions. See *http://www.zeusprod.com/nutshell/links* for a larger list of sound-related Xtras and other URLs of interest.

Audio Xtra (formerly sold as the Sound Xtra)—sound recording at runtime:

http://www.updatestage.com/xtras

DirectSound Xtra from DirectXtras:

http://www.directxtras.com

Speech Recognition—XtrAgent for Windows 95/98/NT support for speech input:

http://www.directxtras.com/xtragent.htm

Multimixer Xtra—extensive control over QuickTime audio tracks:

http://www.turntable.com

Beatnik Xtra (multichannel sound mixing):

http://www.beadspace.com

Din Xtra (unsupported)

The completely unsupported Din Xtra, which has some interesting methods for controlling streaming audio and checking sound channels, is found on the Director 6 CD under:

D:\Macromedia\XDK_d6a4\Goodies\Director\SoundXtr\Xtras\Din.X32

or:

Director 6 CD:Macromedia:XDK for Director 6\Authorware 4:
Goodies:Director:SoundXtr:Xtras:Din

To see the Din Xtra's help text, type this in the Message window:

```
put mMessageList (xtra "Din")
```

Table 15-14 explains the Din Xtra commands. Note that many of these commands appear highly unreliable, and none are officially supported. The Xtra is absent in D7.

Table 15-14: The Unsupported Din Xtra

Din Command	Description/Usage
getChannelCount()	Returns maximum number of sound channels. `set numSoundChannels = getChannelCount()`
getFreeChannel()	Returns number of highest free sound channel: `set highestFreeSoundChannel = getFreeChannel()`

Table 15-14: The Unsupported Din Xtra (continued)

Din Command	Description/Usage
getPlayStatus()	Determines if the sound is playing in specified channel: `set soundPlaying = getPlayStatus (soundChan)` Buggy; seems to always return TRUE. Use *soundBusy()*.
getSndLength()	Returns sound stream's total length in milliseconds: `set length = getSndLength (DinInstance)`
getSndPosition()	Returns sound stream's current time in milliseconds: `set currentTime = getSndPosition (DinInstance)`
pauseRawSound()	Pauses stream played through *playRawSound*: `pauseRawSound (DinInstance)`

playRaw()	Plays the list of files through raw sound as one stream. Returns sound channel:
	`set soundChan = playRaw (DinInstance, ¬`
	`[file1, file2, ...], "aif" | "wav")`

playRegular()	Plays sound through *playSound* mechanism. Returns sound channel:
	`set soundChan = playRegular (DinInstance, ¬`
	`soundFileName, "aif" | "wav")`

setSoundTime()	Sets stream to specified time (in milliseconds): `setSoundTime(DinInstance, time)`
stopSound()	Stops stream played through *playSound* or *playRawSound*: `stopSound (DinInstance)`
stopSoundInChan()	Stops sound in *soundChan*. Returns current time: `set currentTime = stopSoundInChan(soundChan)`

Detecting the Sound Card at Runtime

All Macintosh models should include sound capabilities, but not all Windows PCs do. There are several techniques of varying reliability to detect whether a sound card is installed.

You can try playing a sound and then checking its status. For example, you might play a sound via *puppetSound* (remember to trigger it using *updateStage*) and then check whether *soundBusy()* returns `TRUE` for that channel. You can use a dummy sound containing silence for this test. Make it long enough to give you a chance to check it before it terminates.

Under Windows, you can use *mci* commands as follows:

```
mci "capability waveaudio can play"
set soundCardInstalled = the result
```

But it has been reported that some Windows 3.1 machines return `TRUE` as the result even though they do not have sound cards installed. Furthermore, if the MCI drivers are not properly installed (as is common under Windows 3.1), the result will be `FALSE` even though the system may be sound-capable.

For the highest accuracy, use the Buddy API Xtra's *baSoundInstalled()* method to check for a sound card, or perform a combination of these checks.

MIDI and MCI Device Control

There are a number of ways to play MIDI sounds under Director. The Beatnik Xtra will play MIDI files (and MOD as well).

QuickTime 3 Pro will import a MID file and convert it to a *#midi* track in a QuickTime 3 movie. Any such QT3 movie can be played in D6.5 or D7 via the QT3 Asset Xtra (also requires the QuickTime Musical Instruments extension).

Allegedly, RMI and MID files can be imported as OLE cast members under Windows and played with the MIDI OLE controller.

The Yamaha MIDI Xtra plays back MID files under Windows.

Windows sound cards have a separate MIDI port (configured via the Multimedia Control Panel), which should be independent of the DirectSound/WaveOut device discussed earlier. However, the Yamaha MIDI Xtra will lock out Director sounds. As long as the software synthesizer device is open (even if *soundBusy()* returns `FALSE`), Director can't play sounds via *puppetSound*, for example. Wait for a second (either via Lingo or via extra frames in the Score) after closing the software synthesizer to allow the sound card to switch sound drivers before playing *puppetSounds*.

MIDI files can be played under Windows via appropriate MCI calls. Playing MIDI via MCI is covered in the otherwise outdated "Windows 3.1 Multimedia" once published by QUE, or search Macromedia's TechNotes for the words "MCI" and "MIDI."

You can also use *mci* commands to control WAVE files under Windows (but Director commands won't affect it; for example, *the soundLevel* and *volume of sound* commands won't affect its volume).

Other sound formats

The MOD sound format was designed to store music for video games, especially background music, but is not supported by Director. The MOD Hypercard XCMDs work in D5. In D6 and D7, use the Beatnik Xtra to play MOD files and also RMF (Rich Music Format) files, which are very compact.

Enhanced CDs and RedBook Audio

Director is the dominant application used to create so-called *Enhanced CDs* that combine music and multimedia. *RedBook* is the standard format used for music CDs that are commonly played in home stereo systems. Enhanced CDs (ECDs) known variously or formerly as "CD Extra" or "CD Plus," combine RedBook audio with a separate computer-only (data) session on the same physical disc.

Most CD-ROM burning software, such as Toast, can create Enhanced CDs. Various hardware and software issues ensued before the preferred Enhanced CD format was ironed out a few years ago. The favored format is currently "Stamped Multisession" that conforms to the so-called BlueBook specification, and replaces older approaches such as "Track-Zero."

Cinram (*http://www.cinram.com*) has detailed white papers available on the various CD formats and specifications, including RedBook and BlueBook.

Director can access the RedBook session (via an appropriate Xtra) or the multimedia (data) session, but not both simultaneously.

To access the RedBook audio from within Director requires an Xtra or MCI commands under Windows. To ensure smooth performance, you should not attempt to load multimedia content and play RedBook audio simultaneously or in rapid succession. Here are some possible alternatives:

Load multimedia content into RAM
: If you have sufficient RAM to preload your multimedia content, it can play back from RAM while the RedBook audio is accessed off the CD. The upper limit for RAM playback is probably about 5 to 10 MB.

Load multimedia content onto hard drive
: If the multimedia content is copied to the hard drive, it will not conflict with the attempt to read the RedBook audio from the CD. That said, users don't want large presentations loaded on their hard drive. Keep the content under 20 MB or (preferably) 10 MB. Copying 50 MB of content to someone's hard drive borders on the offensive.

Play RedBook audio at limited times
: Playing RedBook audio throughout your entire presentation might require that the entire presentation to be preloaded or copied to the hard disk. Instead, play RedBook audio only within a small portion of Projector, perhaps via a single jukebox-like interface that can be loaded into RAM. If the remainder of the multimedia content does not require RedBook audio, it can be streamed from the CD as needed.

Use limited non-RedBook Audio
: To simulate RedBook audio being played concurrently with your multimedia content, you can duplicate one or more RedBook tracks as typical Director WAVE or AIFF files on the multimedia session. Most computers don't do justice to CD-quality audio, so you can use 16-bit, 22 kHz, mono tracks or even SWA to save space. You may wish to provide alternate bonus tracks or even music videos instead of mere duplicates of the RedBook tracks available on the album.

Use a caching Xtra
: LRU Cache Xtra (*http://www.mca.com/newumg/lrucache.html)* is designed for making Enhanced CDs. It caches parts of your program so you don't have to copy it to the hard drive.

To calculate how much room will be available on the CD for multimedia content, subtract the size of the RedBook audio from the CD's capacity.

CD-quality audio occupies 176 K/sec or about 10.3 MB per minute. For example, 55 minutes of RedBook audio would require 567 MB of CD-ROM space, leaving

about 100 MB for multimedia. The capacity of CD-ROMs vary from about 650 to 720, depending on the format, manufacturer, and CD-burning software. (See *http://www.cinram.com* for more information.)

ECD resources

Refer to the following ECD resources in addition to the LRU Cache Xtra mentioned earlier.

CD Pro Xtra (free Xtra plays RedBook Audio cross-platform and replaces ECD Control and ECD File):

http://www.penworks.com

Macromedia ECD Control and ECD File XObjects (obsolete toolkit that plays RedBook audio and optionally copies files to hard drive):

http://www.macromedia.com/

ECD mailing list (and links to FAQs):

http://www.turntable.com/ecd/

ECD support web site (includes a database of enhanced CDs):

http://www.musicfan.com/ecd/making.html

Apple Interactive Music Toolkit:

http://www.apple.com

Troubleshooting Sound Problems

If your sound problem is widespread and not exclusive to a particular platform or configuration, then it can usually be addressed by restructuring your Lingo. Unfortunately, many sound problems are configuration-specific, especially under Windows. If the problem occurs under Windows, but not on the Macintosh, refer to the platform-specific caveats under "Sound Mixing Under Windows" earlier in this chapter and in Table 15-1.

Windows sound issues often depend on the sound card or sound driver. Conflicts with RSX and DirectSound are also sometimes reported. When in doubt, obtain the latest device driver for your sound card and the latest versions of RSX and DirectSound if you are using them. In all cases, you should perform compatibility testing on a variety of sound cards with various software installed (with and without RSX and DirectSound). Windows laptops tend to have nonstandard sound cards. Obtain the latest drivers and test on any laptops you are specifically targeting (as for a sales presentation).

Your *ReadMe* file should instruct your Windows users to update their sound card drivers (and RSX and DirectSound if applicable) if they encounter problems.

Sounds may skip during transitions or other processor- or disk-intensive activities. Either preload the sounds or avoid too much concurrent activity, such as loading or streaming other media. Refer to the distinctions between internal and external in Chapter 4 for additional insights.

Here are some common problems that are not specific to a given sound card:

Sounds play in authoring mode, but not in a Projector
: You have most likely omitted the necessary Xtras. See "Xtras needed to play external sounds in Director 6 and 7" earlier in this chapter. You may also have failed to include the external sound files (AIFF, WAVE, or SWA) required by *sound playFile* or linked cast members.

Director asks, "Where is xxxx?"
: You must include external sounds files in the same relative position to your Projector or Director movie as they were during authoring.

Sounds can't be heard at all
: Check *the soundLevel* and *soundEnabled* properties and the volume for the particular item(s) of interest. See Table 15-12. Check the speakers by playing a test sound in the Windows Sound Control Panel.

Sounds drop out in somewhat arbitrary fashion
: In low-memory situations, Director drop outs the sound first. Reduce and optimize memory usage as described in Chapter 9, *Memory and Performance*.

Sound plays too late, particularly under Windows
: If using *puppetSounds*, issue an *updateStage* command to trigger the sound. Playing multiple sounds under Windows introduces a delay (see "Sound mixing latency" earlier in this chapter). Trigger sounds earlier in the Score or premix the audio into one sound using a sound editor.

Sound synchronization is not accurate
: Sound synchronization in Director's Score is never guaranteed, especially when mixing multiple sounds under Windows. Use cue points for improved synchronization and ensure that they are located at the proper points within the audio file (build in some lead time if necessary). Use an audio track in a digital video file for optimal synchronization.

Sounds pop and click or make screeching noises
: Use a clean audio source. Corrupt audio will sound like static or glass shattering and must be replaced or recompressed. Include about 100 to 500 milliseconds of silence at the beginning and end of sounds to reduce popping. RSX and/or DirectSound may also cause popping.

Very short sounds don't play under Windows
: Sounds shorter than 250 milliseconds (1/4 second) may not play at all. Add at least 50 milliseconds of silence to the beginning of the sound and pad the end with some silence if the sound is very short.

Problems playing both QT-based and Director-based sounds simultaneously
: Playing both QuickTime audio and Director audio is not reliably supported under Windows. See "Sound Mixing Under Windows" earlier in this chapter.

Problems switching between QT-based and Director-based sounds
: Make sure that all Director sounds are stopped using *sound stop*, *sound close*, and *puppetSound 0* before playing a QT sound. Ensure that QT sound is stopped by setting *the movieRate of sprite* to 0 or setting *the sound of member* to `FALSE` before playing a non-QT sound. Allow Director to release the sound device by setting *the soundKeepDevice* to `FALSE`.

A digital video's audio can't be heard even though no other sounds are playing

Ensure that the video contains an audio track and that *the sound of member* is `TRUE` and *the volume of sprite* is 255. Ensure that *the movieRate of sprite* is 1. Check *the soundLevel* property. Under Windows, you may need to stop all non-QuickTime sounds to free the sound card for QuickTime usage if not using QT3Mix.

Multiple audio tracks in a digital video cannot be heard

QuickTime 2 for Windows can't handle more than one audio track inside a single QuickTime movie. If necessary, separate the excess audio tracks into distinct audio-only QT movies. QuickTime 3 should handle multiple tracks within a single QuickTime movie and multiple QuickTime movies with separate audio tracks.

Sounds play at the wrong pitch (chipmunk-like or very low-pitched)

If using a non-standard rate, such as 15 KHz, Director will resample audio to the closest rate that the sound card supports. Use the standard Windows sampling rates: 11.025, 22.050, or 44.100 kHz.

Last few seconds of SWA or MP3 don't play

An SWA won't play the final few seconds of its sound if there is a cue point at the very end of the file. Use an alternate method for timing near the end of an SWA stream, such as checking *the currentTime of sprite* manually. The last few seconds of an MP3 file won't play in D7.

Can't import AIFF, WAVE, or SWA files

The MIX Services and Sound Import Export Xtras are needed to import sound files both during authoring and at runtime. Director won't import some compressed WAVE files, depending on the type of compression.

Tempo channel Wait setting interferes with interactivity

In Director 5, the Tempo channel locked out interactivity. In Director 6, using the *Wait for Cue Point* option may prevent Custom Buttons from behaving as expected. In either case, use Lingo alternatives to the Tempo channel, such as manually checking *soundBusy()* or waiting for *cuePassed* events as shown in Examples 15-4 and 15-6.

Exported QT or AVI files missing sounds

Sound tracks are often omitted during QuickTime export. Even if exported successfully, Director sometimes creates an excessive number of separate audio tracks in the QuickTime movie. Add the sound track to the exported digital video with a separate QuickTime editing application.

Sound Editing Applications and Utilities

Many applications can be used to create Director-ready sound files in either AIFF or WAVE format. SWA files can be created via SoundEdit, Peak LE, or Director for Windows.

SoundEdit is made by Macromedia and is included with Director 6 Multimedia Studio for the Macintosh (see also the SoundEdit Automator and SWAtomator utilities that come with it):

http://www.macromedia.com/support/soundedit/

Peak LE is included with the Director 7 Multimedia Studio for the Macintosh. The full version is sold by Bias Inc., which also sells Sound Designer:

http://www.bias-inc.com

Sound Forge XP (included with Director Multimedia Studio for Windows) is made by Sonic Foundry:

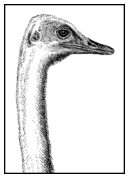
http://www.sfoundry.com

Cool Edit 96 shareware and professional versions (Windows only):

http://www.syntrillium.com

BarbaBatch by MacSourcery (Mac only) converts batches of sounds between various formats:

http://www.macsourcery.com/web/BarbaBatch/barbabatch.html



CHAPTER 16

Digital Video

Director is not a video editing tool. Although you can make minor edits to DV files within Director, you will ordinarily create your DV files in an external application such as Adobe Premiere and then import them into Director. See "Applications and Tools" later in this chapter and *http://www.zeusprod.com/nutshell/dvtools.html* for additional information on external digital-video related applications.

Digital Video in Director

The term "movie" is used to indicate Director movies (DIR files), QuickTime movies (MOV files), Video for Windows movies (AVI files), and also Movies-in-a-Window. My use of the word "movie" in this chapter should be clear from context, but be explicit when asking for technical assistance.

AVI is a file format that is supported by Video for Windows (VFW), or its successors, ActiveMovie and DirectShow. Director supports only the Video for Windows API. The DirectMedia Xtra (*http://tbaiana.com*) can take advantage of DirectShow. This chapter focuses primarily on QuickTime, although most of it applies to AVI files as well. For information on AVI, VFW, and ActiveMovie, see *http://www.mmii.com/directorhelp/avi01.htm* and also *http://camars.kaist.ac.kr/~jaewon/special/avi/avi.html*.

As of February 1999, QT4 has not yet shipped. There are no major architectural changes between QT3 and QT4 on either platform, so D7 should work with QT4 using the QT3 Asset Xtra, as it does with QT3.

Digital Video Cast Members

Director supports QuickTime on the Macintosh and both QuickTime for Windows (QTW) and Audio Video Interleave (AVI) under Windows, referred to collectively as digital video (DV). The QuickTime 3 Asset Xtra adds QuickTime 3 (QT3) support to Director 6.5 and 7 on both Macintosh and Windows. Director 7

supports QT3 and QT4, but no longer supports QT2.x. Versions prior to Director 6.5 supported QT 2.x or earlier only.

Note that there are two distinct types of DV cast members: old *#digitalVideo* members and newer *#quickTimeMedia* members. DV files imported via `File ➤ Import` create *#digitalVideo* (QT2 or AVI) members in D6. DV files inserted via `Insert ➤ Media Element ➤ QuickTime 3` or via `File ➤ Import` in D7 ordinarily create *#quickTimeMedia* (QT3) members (but AVI files under Windows can be imported as either *#digitalVideo* or *#quickTimeMedia* members in D7). All DV files are always externally linked.

DV cast members use the first frame of the external video file as their thumbnail, and are distinguished by a small video camera or QuickTime 3 icon as shown in Figure 4-3.

Sprites in the Score and Playback on the Stage

DV sprites can be placed in any sprite channel. When Director is not running, the first video frame of the DV cast member is displayed on the Stage, even if the DV sprite spans multiple Score frames. The DV sprite will not update or play until the Score's playback head is moving.

A DV sprite will play at the frame rate intrinsic to the digital video file (see Example 16-4) unless overridden by the commands in Tables 16-12 and 16-13. A video's playback rate is not affected by the Tempo channel's frame rate setting or the *puppetTempo* command.

When a DV sprite is tweened out over multiple Score frames, one Score frame will *not* correspond to one frame of the digital video. See Chapter 1, *How Director Works*, for a discussion of Director's frame-based animation model, and Chapter 3, *The Score and Animation*, for details on the Tempo channel.

To wait for a DV sprite to reach a particular cue point, use the Tempo channel's *Wait for Cue Point* option. Indicate the channel containing the DV sprite of interest and choose from the list of available cue points within the DV, or `{Next}` or `{End}`. Use Lingo instead for more flexible and powerful control of DV sprites (see Tables 16-13 and 16-16).

When a DV sprite is played direct-to-Stage, it appears in the foremost paint layer, using the Copy ink effect, and leaves trails. Director does not automatically refresh the Stage area within the DV sprite's bounding rectangle. You must force a refresh by moving a non-DV sprite over the affected area, by performing a transition, or using *set the stageColor = the stageColor*.

Compression and Decompression (Codecs)

The *uncompressed* size of a single video track, in bytes, can be calculated as:

```
(the width of member) × (the height of member) ×
(bits per pixel/8) × frames per second × duration in seconds
```

This calculation does not include other tracks in a digital video file, such as secondary video tracks, multiple sound tracks, text tracks, and so on. For example, uncompressed CD-quality audio adds 176 K/sec to a digital video's data rate.

Because uncompressed video requires too much bandwidth, video is usually compressed with an appropriate algorithm called a *codec* (COmpressor-DECompressor). The DV file's *compressed* data rate is determined by the nature of the source material and the chosen codec, output file's dimensions, bit-depth, frame rate, keyframe interval, and output quality (lossless or lossy).

Use Media Cleaner Pro (*http://www.terran-int.com*) or a similar utility to compress your video. A typical compression ratio might be 9:1 for Cinepak, 50:1 for Sorenson, and possibly 100:1 for animation compression assuming clean source. QuickTime also supports compressed sound tracks, but the audio bandwidth is usually minor compared to the video bandwidth.

The MPEG and DirectMedia Xtras (*http://ww.tbaiana.com*) play back MPEG-1 and MPEG-2 compressed video.

A full discussion of codecs is beyond the scope of this book. See the information at *http://www.zeusprod.com/quicktime/codecs.html* for a list of codecs supported by QT2 and QT3 and a comparison of different codecs.

If you are using a proprietary Xtra, codec, or engine for video playback, contact the manufacturer for licensing and distribution issues.

Digital Video Performance

The CPU, available RAM, drive performance, video card, VRAM, video driver, OS version, QuickTime or Video for Windows version, monitor depth, and software configuration all affect digital video performance.

DV data is streamed from disk as it is played, enabling a large file to be played without requiring excessive RAM (although SW7 and QT3 don't support Internet streaming, QT4 might). The video and audio tracks are interleaved in a DV file, so that they can be read in quick succession as time passes. (Improper interleaving of audio and video severely degrade DV playback.) If the disk's transfer rate cannot keep up with the video file's data rate, QuickTime or Video for Windows will skip video data to maintain sound synchronization, unless the *Play Every Frame (No Sound)* option is set. This will cause the video to appear jerky and in extreme cases may cause the audio to drop out.

All else being equal, DV performance is affected primarily by its data rate.

A video's *average* data rate (in KB/second) can be determined by:

```
(size of the external data file in KB) ×
float(the duration of member) / (the digitalVideoTimeScale)
```

(See Example 4-6, which determines the size of an external file.) A video's *peak* data rate is sometimes of more concern, although the peak data rate should not be significantly higher than the average data rate if the movie is properly prepared. Table 9-3 lists acceptable video data rates for various speed CD-ROM drives.

Use Adobe Premiere or a similar utility to determine a movie's peak data rate or detect improper interleaving.

Playing DV sprites non-direct-to-Stage or stretching DV sprites in increments other than 100% is *very* slow. Avoid moving a video sprite while it is playing—use a still frame from the video when performing animations or transitions. Place DV sprites at an offset that is evenly divisible by 8 relative to the upper left of the monitor for improved performance. Prior to D7, the upper-left coordinate of the Stage should always be an even multiple of 8, so position your sprite such that its upper-left edge is at an offset of 160 pixels (for example), not 159 pixels. In D7, the Stage is no longer constrained to offsets that are multiples of 8, but you should position the Stage at an even multiple anyway. The width and height of the external DV file (and the DV sprite if stretched) should also be evenly divisible by 8, such as 320×240 pixels.

Even if a DV file performs adequately in MoviePlayer, it may lag within Director due to additional overhead or suboptimal Score or Lingo techniques. Minimize other activities, such as loading or animating large cast members or playing additional sounds or video while playing a DV sprite. If applicable, interleave your audio and video within a single DV file, rather than attempting to play separate audio and video files simultaneously. Test DV performance early in the design process within a realistic Director prototype (not simply in isolation) on all supported platforms.

Preloading digital video

The preLoad of member property (equivalent to the *Enable Preload* option in the DV Properties dialog box) determines whether a digital video's data is preloaded into RAM before it is played. The amount of RAM used for preloading is set by *the preLoadRAM* (which is a system-wide property, not a cast member property).

I'm not convinced that preloading DV (often called *pre-rolling*) has ever worked correctly in Director or that it would be terribly useful if it did. If the DV data rate is sufficiently low, there should be no need to preload the data; it should be provided as needed on the fly. Preloading is only of practical use for very small videos that must play smoothly with no frames dropped, such as a simulated visual transition or animation.

Preloading entire videos can cause a long delay and consume excessive RAM. Preloading a portion of a video may result in a noticeable "hiccup" when the preloaded data runs out. In most cases, you are better off preloading other assets instead. For example, to avoid accessing a CD-ROM in two places at once, preload any animation or audio, and stream the video from disk as usual.

If *the preLoad of member* is `TRUE`, the default setting for *the preLoadRAM* (0) uses all available memory for preloading.

If you insist on preloading digital video, you can specify a fixed amount of RAM, such as 2 MB:

```
set the preLoadRAM = 2 * 1024 * 1024 -- 2 MB
```

Or use a percentage of *the freeBlock*, such as:

```
set the preLoadRAM = 0.5 * the freeBlock
```

To avoid the characteristic hiccupping of a QT movie when it first starts, include about a 0.5 second lead-in to your video track (perhaps some extra black frames followed by a dissolve). A second technique is to load the cast member header before the video data is needed. Set *the pausedAtStart of member* to `TRUE` and use *the preLoad member* command to preload the header information (or place the DV sprite off-Stage before the frame in which it is needed). Set *the movieRate of sprite* to 1 to start the video in the following frame of the Score. If hiccupping remains evident, add a few extra Score frames between the frame in which the video sprite first occurs and the frame in which you start it playing.

The purgePriority of member property (equivalent to the *Unload* option in the *DV Properties* dialog box) is largely irrelevant for DV cast members, because their data is always unloaded immediately after playback. Only the header information (several hundred bytes) is unloaded based on this property.

QuickTime 2, QuickTime 3, and QuickTime 3 Pro

Prior to QuickTime 3, the latest Macintosh release was QT 2.5 and the latest Windows release was QTW 2.1.2 (referred to here collectively as *QT2*). Director 6.5 and earlier for both Macintosh and Windows support QT2 (as long as the system software is installed) without requiring any Xtras.

The QuickTime 3 Asset Xtra (see Table 16-3) is required to use QT3 (*#quickTimeMedia*) members in D6.5 and D7. Director 6.0.x does not support QT3.

Apple's QT3 Extension supports all Macintoshes, but Macromedia's QT3 Xtra requires a PowerMac (or G3). On the Macintosh, the QT3 extension replaces QT2.x entirely; the two versions can not cohabitate. In D6.x, existing Macintosh QT2-style (*#digitalVideo*) cast members will work whether QT2 or QT3 is installed. The Macintosh QT3 System Extension and QT3 Xtra add support for QT3-specific features, such as importing QTVR, QD3D, and AVI files.

Director 7 supports only QT3 members. QT2-style *#digitalVideo* members are converted to *#quickTimeMedia* members when updating DIR files from D6 to D7.

QTW3 requires Windows 95/98/NT and does not support Windows 3.1. Windows 95/98/NT support *simultaneous* installation of up to 3 versions of QTW—16-bit and 32-bit versions of QTW2, plus the 32-bit QTW3. QTW3 has a completely new architecture and is not backward-compatible with QTW2 cast members under Windows. Therefore, older *#digitalVideo* cast members won't play unless QTW2 is installed, regardless of whether QTW3 is installed. Windows 3.1 supports only the older 16-bit version of QuickTime for Windows (QTW 2.1.2). As on the Macintosh, D7 for

Windows supports only QT3 cast members, and the *#digitalVideo* type is supported only for AVI files in D7.

Table 16-1 lists the versions of QT supported on each platform for Director 6.5 and Shockwave 6.0.1. D7 and SW7 support only QT3 and D6.0 supports only QT2.

Table 16-1: Supported QuickTime Versions in D6.5

Platform	Supported QT Versions in D6.5	D6.X #digitalVideo Members Require	D6.5 and D7 #quickTimeMedia Members Require
Macintosh 68K	QT2 or QT3	QT2 or QT3	Never supported
PowerMac/G3	QT2 or QT3	QT2 or QT3	QT3 and QT3 Xtra
Windows 3.1	16-bit QTW2 only	QTW2	Never supported
Windows 95/98/NT	16-bit QTW2, 32-bit QTW2, QTW3 concurrently installed	16-bit or 32-bit QTW2 matching Projector	QTW3 and QT3 Xtra
Shockwave movie	QT2 (SW6); QT3 (SW7)	SW6[1]	QT3 and QT3 Xtra[1]
Browser outside Shockwave	QT2 and QT3	QT2 browser plug-in	QT3 browser plug-in[2]

[1] Shockwave 6 does not require an Xtra or plug-in to play QT2 video. It requires the same QT system software as is needed by a Projector on that platform. Likewise, QT3-style cast members require the same Xtras as Projectors, but placed in the Shockwave *Xtras* folder. QuickTime video doesn't stream in Shockwave. It must be downloaded in its entirety first.
[2] The QuickTime 3 browser plug-in is installed by the QT3 Installer from Apple and also comes with the major browsers.

Differences in QT2 and QT3 capabilities

QT3 cast members support new features in Director, including:

- Many new file formats, including QTVR 2.0, QD3D, AVI, and GIF.
- New codecs, most notably the Sorenson codec (*http://www.s-vision.com*) which yields better compression than Cinepak in most cases.
- Loop points within QuickTime media.
- A 1-bit mask cast member can be used as a non-rectangular mask for direct-to-Stage video.
- Ability to rotate, scale, and offset QuickTime media within a sprite's bounding rectangle.
- Non direct-to-Stage playback under Windows 95/98/NT in addition to the Macintosh (although it's not recommended, for performance reasons).
- Under Windows, Director-based and QT-based sounds can be mixed if the system is properly configured. See "Sound Mixing Under Windows" in Chapter 15.
- QT3 itself offers special effects and transitions within QT movies. These must be added in a third-party video editor such as Adobe Premiere before the video is inserted into Director's cast.

Differences in QT2 and QT3 usage in Director

You must use QT2 cast members to retain support for Windows 3.1 and 68K Macintoshes, as neither are supported by the QT3 Xtra.

To take full advantage of QT3 on both platforms, QT3 cast members must be imported via `Insert ➤ Media Element ➤ QuickTime 3`, instead of imported via `File ➤ Import`. Unlike `File ➤ Import`, Director 6.5's `Insert` function hard codes the path to external files. Manually substitute in the `@` relative path operator for the current folder in the *QT3 Cast Member Properties* dialog box. See Chapter 4, *CastLibs, Cast Members, and Sprites*.

In D7, `File ➤ Import` will import QT3 members on both platforms. QT3 uses the *mRate*, *mTime*, and *volumeLevel of sprite* properties instead of the *movieRate*, *movieTime*, and *volume of sprite* properties. Throughout this chapter, when I refer to the *movieTime* and *movieRate* properties, it is implicit that you should use *the mTime of sprite* and *mRate of sprite* properties for QT3 sprites in D6.5.

QT3 can read QT2 movie files, but not vice versa. However, some QT2 movies may display artifacts when played with QT3. Resave files in QT3 format with the QT3 Movie Player for best results.

Under Windows, QTW3 is much more processor-intensive; it may drop audio and video during playback on machines on which QTW 2.1.2 played well, such as Pentiums below 133 MHz. The performance hit occurs both inside Director and in the standalone MoviePlayer. QTW3 performs noticeably better under Windows if DirectX is installed/enabled. DirectX can be obtained online from *http://www.microsoft.com/directx*.

Determining the Installed Digital Video Software

Ideally, you will include an installer that guarantees that the necessary version(s) of QuickTime or Video For Windows are installed. Regardless, you should check for the presence of the required DV software via Lingo and warn the user or run an installer if it is not available.

Detecting Video for Windows

Example 16-1 detects the presence of Video for Windows (VFW) and its version. It returns `EMPTY` is VFW is not installed and returns a text string containing the version, such as "4.0.95", if VFW is installed.

Example 16-1: Checking the VFW Version

```
on getVFWversion
  if the videoForWindowsPresent then
    MCI "open AVIVIDEO alias filename.avi"
    MCI "info filename.avi version"
    return the result
  else
     return EMPTY
  end if
end getVFWversion
```

As not all Windows system are MCI-enabled, you can also use the Buddy API Xtra's *baVersion()* method:

```
put baVersion("vfw")
-- "4.0.95"
```

Detecting QuickTime 2 and QuickTime 3

To detect the QuickTime's version and presence, use the *quickTimeVersion()* method (requires the QT3 Asset Xtra) and *the quickTimePresent* property. These indicators vary markedly across platforms and in D7, as shown in Example 16-2 and Table 16-2.

The number of digits displayed of the value return from *quickTimeVersion()* depends on *the floatPrecision*.

Example 16-2 sets two separate global variables: *gHasQT3*, indicating whether QT3 is installed; and *gHasQT2*, indicating whether QT2 (or prior) is installed—and is necessary only in D6.5. It checks for the presence of the QT3 Xtra before calling *quickTimeVersion()*. See the following sections for an explanation of its pretzel-like contortions. In D7, the QT2 is never supported and *the quickTimePresent* indicates whether QT3 is installed.

Example 16-2: Checking for QT2 and QT3 Presence in D6.5

```
on checkQTversions
  global gHasQT2, gHasQT3
  if integer (char 1 of the productVersion) >= 7 then
    set gHasQT3 = the quickTimePresent
    set gHasQT2 = FALSE
    exit
  endif
  --Remainder of code pertains to D6.5
  if the platform contains "Windows" then
    -- Windows may have both QT2 and QT3!
    set gHasQT3 = (quickTime3check() >= 3.0)
    set gHasQT2 = the quickTimePresent
  else
    -- Macintosh has only one version of QT installed
    if the quickTimePresent then
      set gHasQT3 = (quickTime3check() >= 3.0)
      set gHasQT2 = not (gHasQT3)
    else
      set gHasQT3 = FALSE
      set gHasQT2 = FALSE
    end if
  end if
end checkQTversions

on quickTime3check
  -- This checks whether the QT3 Xtra is installed
  -- before attempting to call quickTimeVersion()
  repeat with x = 1 to the number of Xtras
    if the name of xtra x = "QuickTimeSupport" then
```

Example 16-2: Checking for QT2 and QT3 Presence in D6.5 (continued)

```
      return quickTimeVersion()
    end if
  end repeat
  if the quickTimePresent then
    -- Assumes QT 2.x is installed if pre-QT3 version is found
    return 2.0
  else
    return 0
  end if
end
```

To determine the installed browser QT plug-in, you must use JavaScript.

Detecting QuickTime 2 or QuickTime 3 on the Macintosh

In D6 and earlier versions on the Macintosh, *the quickTimePresent* indicates whether any version of the QuickTime Extension (plus the QuickTime PowerPlug on PowerPCs) is installed in the *System:Extensions* folder. In D7, it returns `TRUE` only if QT3 or higher is installed.

The *quickTimeVersion()* function returns the QT version if QT3 or higher is installed. If not, *quickTimeVersion()* returns a –2147483648 error during authoring. Furthermore, if QT3 or higher is not installed, the QT3 Asset Xtra for Projectors won't load, and calling *quickTimeVersion()* causes a "Handler not defined" error.

If the QuickTime Extension but not the QuickTime PowerPlug is installed, MoviePlayer will still function, but Director will not allow you to import, insert, or open QT cast members. When changes are made to the QuickTime Extensions, the Macintosh must be rebooted before they take effect.

Unlike the *quickTimeVersion()* function on the Macintosh, the Buddy API Xtra's *baVersion()* method won't fail if QT3 is not installed:

```
put baVersion("qt")
-- "3.0"
```

Detecting QuickTime 2 and QuickTime 3 under Windows

In D6 and D6.5 under Windows, *the quickTimePresent* indicates whether the version of QTW2 (16-bit or 32-bit) matching the Projector is installed. (It looks for either *QTW.DLL* or *QTW32.DLL* in the *Windows\System* folder.) In D7, *the quickTimePresent* is `TRUE` only if QTW3 is installed. *QuickTimeVersion()* checks for the necessary QTW components, including *QuickTime.qts* in the *Windows\System* folder.

You must restart Director, but not Windows, to make it recognize any changes to the QuickTime installation. The QTW3 uninstaller provided by Apple unregisters the QTW3 components, but deletes only those items in the *C:\Program Files\QuickTime* folder, not the actual QTW3 software drivers (specifically *QuickTime.qts*) in the *Windows\System* folder. Therefore, even after an uninstall, *quickTimeVersion()* will report that QTW3 is still installed. To perform a complete "uninstall," use the QTW3 uninstaller, then delete all the files listed at *http://www.zeusprod.com/quicktime/qtfiles.html*. (The QTW3 control panel doesn't include a list of installed component files as did QTW2.x.)

Table 16-2 explains testing *the quickTimePresent* and *quickTimeVersion()* under Windows.

Table 16-2: The QuickTimePresent and QuickTimeVersion()

Projector	16-bit QTW2	32-bit QTW2	QTW3	quickTimePresent		quickTimeVersion()[1]
				D6	D7	
16-bit		N/A	N/A	FALSE	N/A	Error[2]
16-bit	✓	N/A	N/A	TRUE	N/A	Error
32-bit	N/A			FALSE	FALSE	0.0000
32-bit	N/A	✓		TRUE	FALSE	2.0000,[3] 2.1200, etc.
32-bit	N/A		✓	FALSE	TRUE	3.0000, 3.0100, 3.0200
32-bit	N/A	✓	✓	TRUE	TRUE	3.0000, 3.0100, 3.0200

[1] The number of digits of the returned value displayed depends on *the floatPrecision*.
[2] The QuickTime Asset.X32 Xtra doesn't load when running a 16-bit projector under any version of Windows. There is no .X16 (16-bit) version.
[3] The *quickTimeVersion( )* method reports 2.000 for any versions of QTW prior to version 2.0.

Example 16-3 detects QTW2's presence and version. It returns `EMPTY` if QTW2 is not installed and returns a text string containing the version, such as "2.11", if QTW2 is installed. It does not recognize or check for QTW3.

Example 16-3: Checking the QTW2 Version

```
on getQTW2version
  MCI "open QTWVIDEO alias filename.mov"
  MCI "info filename.mov version"
  return the result
end getQTW2version
```

You can also use the Buddy API Xtra's *baVersion()* method to check the QTW2 and QTW3 versions separately. It returns a different string than the MCI call:

```
put baVersion("qt")
-- "2.1.1.50 Beta 1"
put baVersion("qt3")
-- "3.0.2"
```

The QuickTime 3 Asset Xtra

Table 16-3 lists the QT3 Xtras you'll need for authoring and distribution. The authoring-time QuickTime Asset Options version of the Xtra will prevent a Projector from launching; distribute the QuickTime Asset instead. Windows 3.1 and 68K are not supported by the QT3 Xtra. Director D6.5 or D7 is required when using the QT3 Xtras.

The QuickTime Asset Xtra shipped with Projectors will not load on PowerMacs unless QT3 is installed, causing an error when Director encounters movies using *#quickTimeMedia* cast members.

To avoid an error message, delete the QuickTime 3 Xtra from the list under `Modify ➤ Movie ➤ Xtras` before saving each movie using *#quickTimeMedia* members. Don't bundle the movie into the Projector. Check whether the QuickTime 3 Xtra is available as shown in Example 16-2 before using a Director movie or castLib containing *#quickTimeMedia* members.

Table 16-3: QuickTime 3 Xtras

Projectors	**Xtra Name**
Power Macintosh Authoring	QuickTime Asset Options PPC (D6.5) QuickTime Asset Options (D7)
Power Macintosh Distribution (Projectors or Shockwave)	QuickTime Asset PPC (D6.5) QuickTime Asset (D7)
Windows 32-bit Authoring	QuickTime Asset Options.X32 (D6.5) QTAuth.X32 (D7) QTExport.X32 (D7)
Windows 32-bit Distribution (Projectors or Shockwave)	QuickTime Asset.X32 (D6.5) QT3Asset.X32 (D7)

The QT3 Xtra requires at least 15 MB allocated to Director on the PPC. Although Projectors seem to work with the default allocation (less than 7.5 MB), you should consider increasing it when using QT3.

The QT3 Asset Xtra can import many data types, including QTVR 2.0 and QD3D. QTVR 1.0 files may not work with the QT3 Asset Xtra. Update your QTVR files to QTVR 2.0 or higher. On the Macintosh, Macromedia recommends deleting all old versions of QTVR and QT-related extensions and doing a clean install of QT3.

In D6.5, the QT3 Asset Xtra does share CPU cycles with Director to allow sufficient time to animate other sprites while a QTVR sprite is playing direct-to-Stage. The D7 version of the Xtra is a bit friendlier.

The QT3 Asset Xtra provides new Lingo commands and properties, but all older digital video properties also work with media inserted as QuickTime 3 cast members. See *Show Me 6_5/QT3/qt3_showme.dir*, which comes with D6.5. Also see Macromedia's web site.

Note that QuickTime 3 itself implements many features that are not supported fully by Macromedia's QT3 Xtra.

Cross-Platform Digital Video Issues

QuickTime allows you to use a single QT movie (MOV) file on both Macintosh and Windows. To use QT2 movie files under Windows, simply flatten and de-fork them by saving them with the *Self-Contained* and *Playable on non-Apple Computers* options with MoviePlayer (other DV editing software has similar options). This collects all DV data into a single file and removes the resource fork (such files will still play on the Macintosh). See Macromedia TechNote #12113, "Supported Digital Video Formats." All QT3 movies are cross-platform by design, and for performance reasons they should be self-contained.

There are some substantial platform differences pertaining to DV, as shown in Table 16-4. See Table 16-5 for more details on importing and exporting.

Table 16-4: Cross-Platform Digital Video Differences

Feature	**Macintosh**	**Windows**
Video formats for import	QT2 (.MOV, 'MooV'); D6.5 or later imports QT3, VFW (.AVI, 'VfW '),[1] and MPEG, .MPG ('MPEG').[2]	QTW2 (.MOV),[3] VFW (.AVI), QT3 Xtra (D6.5) imports QTW3 formats, but not MPEG.
Video formats for export	QT2 (prior to D7); QT3 (D7 or later).	VFW (.AVI).
Direct-to-Stage video	Optional for QT2 and QT3.	Optional for VFW and QTW3. Mandatory for QTW2.
Standard video controller	Optional for QT2 and QT3.	Optional for QTW, not supported for AVI files.
QuickTime movie format	Allows file dependencies and resource fork.	QT2 movies must be flattened and de-forked. QT3 allows file dependencies.
QuickTime software	Extension is included with Mac OS, but can be disabled.	Must install QTW version to match Projector. (16-bit or 32-bit). D7 requires QTW3 (32-bit).
the timeScale of member property	Defaults to 600 for QuickTime.	Defaults to 30 for QTW and to 60 for VFW.
Multiple audio track support	Yes for QT2 and QT3.	No for AVI and QTW2; yes for QTW3.
Simultaneous Director and DV audio	Yes.	Not always. See "Sound Mixing under Windows" in Chapter 15.
Different custom palettes for multiple digital videos	QT2 requires Fix Palette Xobject to reset palette between movies. QT3 does not.	Does not require any special accommodations.

[1] An AVI (digitalVideoType *#videoForWindows*) cast member imported via D6 for Windows will be converted to a QT cast member (digitalVideoType *#quickTime*) on the Macintosh if QT3 is installed. In D7, an AVI member will be converted to a *#quickTimeMedia* member on the Macintosh if QT3 is installed. If QT2.x only is installed, Director will issue a –2048 error when trying to use the AVI member on the Macintosh.
[2] MPEG import requires the QuickTime MPEG extension.
[3] QT2 files must be "flattened and deforked" for Windows playback.

Once you have settled on your DV format, bear in mind:

- Non-direct-to-Stage video is not always cross-platform-compatible (nor recommended). See Table 16-7.
- The same video with the same palette may appear slightly darker under Windows than on the Macintosh. Test under both platforms to avoid problems.
- Video performance is rarely the same on different machines and different platforms. Test on a variety of machines, including ones with different video cards, to detect any problems.
- Proprietary video playback schemes may involve a runtime royalty fee, may not be cross-platform, and may not work as advertised.
- Video for Windows is not fully supported on the Macintosh (although AVI files can be played via QuickTime 3).
- Sound and transitions are not always exported when using QT export. Add the sound track to the exported digital video in a separate video editing tool. Add transitions in a separate tool such as Adobe Premiere.

Installation and Licensing Issues

QuickTime 3 can be be downloaded for free (*http://www.apple.com/quicktime*) and is also available for $10 on a CD.

The QT3 installer will create a folder named *QuickTime Folder* in the root of the System disk on the Macintosh, and named *C:\Program Files\QuickTime* under Windows. That folder includes the MoviePlayer and PictureViewer utilities and some sample files, but the QuickTime software itself is installed in the Macintosh *Extensions* folder or *C:\Windows\System* directory. See *http://www.zeusprod.com/quicktime/qtfiles.html* for a complete list of files and where they are installed.

The *Get QuickTime Pro* movie is a special advertisement that is copied to your desktop by the QT3 installer. It does not behave properly in Director. Use the *Sample* movie installed in the *QuickTime* folder if you need a test video to debug QT3 problems.

QT3 Pro is not a different version of QuickTime; it is an enhancement to QT3 that adds the ability to import and export multiple formats from MoviePlayer and PictureViewer. Upgrade to QT3 3 Pro ($30 per platform) via the *Get QuickTime Pro* demo movie, via the *Registration* option in the QuickTime 3 Control Panel, or at:

http://www.apple.com/quicktime/rights/

Macintosh users usually have the QuickTime extension installed, but you should provide a QT3 installer if requiring QT3. Many but certainly not all Windows users have QTW installed, so you should license and distribute the Apple QTW installer (currently free). At least until QTW4 is released, Apple licenses both the QTW3 and older QTW2.1.2 installers.

You can install the older 16-bit QTW 2.1.2 under Windows 3.1. You can optionally QTW 2.1.2 in addition to QTW3 on Windows 95/98/NT systems.

If creating a custom installer for other components, have it launch the Apple QT installer. (Most commercial installers will check for specified files on disk or entries

in the Windows Registry file to determine whether components need to be installed.)

A Macintosh must be rebooted after installing QT; Windows does not need to be rebooted, but you must restart Director or your Projector for it to recognize the updated QT installation. The QTW installer cannot install QTW while a Projector using QTW is running. Use zLaunch (*http://www.zeusprod.com/products/zlaunch.html*) to quit and restart a Projector after QTW is installed. You *must* use the Apple QT3 Installer in its entirety if you intend to install any QT components. Apple's QT3 installer will check which versions of QuickTime are already installed, and install only what is needed.

When using AVI files, you may want to provide an installer. VFW is included with Windows 95, but some Windows 3.1 users may not have it. VFW's successors, ActiveMovie or DirectShow, are included with Win95 OSR2, Win98, and WinNT, so all of these can play AVI files. You can distribute the VFW installer from the Director 6 CD if necessary. The DirectMedia Xtra (*http://www.tbaiana.com*) will check if DirectShow is installed (see its *isDirectShowInstalled()* function).

QT2 or QT3 can be played in a browser without Shockwave if the user has the Netscape-compatible QT2 or QT3 plug-in installed (and the appropriate version of QT installed). Shockwave 6, like D6 Projectors, can play QT2 content without an Xtra, but QT3-specific content requires the QT3 Xtra (see Tables 16-1, 16-2, and 16-3).

Licensing

Apple licenses the basic QT installer for Macintosh and Windows for free, assuming the installer displays their *Get QuickTime Pro* advertisement, but you must fill out a licensing agreement. For an additional fee, you can distribute QT3 without the advertisement or even QT3 Pro, but there is rarely a need for this in Director.

The current QuickTime Licensing terms are available at:

http://gemma.apple.com/mkt/registering/swl/agreements.html
http://developer.apple.com/mkt/registering/swl/qtannouncement.html

To license the QT installer(s), download the PDF file:

http://developer.apple.com/mkt/registering/swl/agreements.html#QuickTime

Print out the PDF file using Acrobat Reader, fill it out, and mail signed copies to:

Apple Computer, Inc.
Software Licensing M/S 198-SWL
2420 Ridgepoint Drive, Austin, TX 78754

Follow up an email to *sw.license@apple.com* or with a phone call to 800-793-9378 or 512-919-2645 and ask for an FTP address from which to download the file once they have received your contract. The licensed installer includes a *QTSETUP.INI* file that can customize the installation prompts somewhat.

User Interface Issues

DV sprites may affect your product design. Bear in mind the following:

- DV sprites are ordinarily played direct-to-Stage, which ignores any ink effects (Copy ink is used) and displays the DV sprite in front of all other sprites regardless of their sprite channel numbers.
- The cursor will flicker when it passes over a DV sprite. Either hide the cursor or tolerate the flicker.
- Transitions are very slow under Windows when a DV sprite is playing. Stop the DV sprite before performing a transition. Do not attempt to use transitions with other sprites while a DV sprite is playing; any transition will include the DV sprite's bounding rectangle, because a DV sprite is always changing. Either bring the other sprites in from off-Stage or imitate a transition with a series of cast members or by using a blend ink (see Example 13-5).
- DV sprites will play back using the custom palette in effect, but certain caveats apply. Refer to Chapter 13, *Graphics, Color, and Palettes,* for details on palette management.

Sound Issues

The Macintosh has multiple sound channels, so you can play audio tracks within QT files simultaneously with Director-based sounds (AIFF, WAVE, SWA, and internal sounds).

Windows has only one true sound channel, and QuickTime for Windows and Video for Windows seize it when playing audio tracks within DV files. This may prevent Director from playing Director-based sounds while a DV file with an audio track is playing. Director's MacroMix technology mixes Director-based sounds together to simulate multichannel audio, and it too seizes the sole Windows sound channel when it plays a sound.

Whichever component takes control of the sound channel first may lock out the other from playing sounds. D6.5 or D7 in conjunction with QT3 can sometimes mix QT-based and Director-based sounds simultaneously. See "Sound Mixing under Windows" in Chapter 15 for details on avoiding sound conflicts.

Digital Video Tools and Options

This section describes the digital video cast member properties and user interface options.

Video Playback and Editing Window

Use Director's Digital Video window, shown in Figure 16-1, to preview or perform simple edits on DV cast members. Don't confuse the Digital Video window with a DV sprite placed on the Stage, which does not have its own window, or with movies-in-a-window (MIAWs) which are Director files, not digital videos.

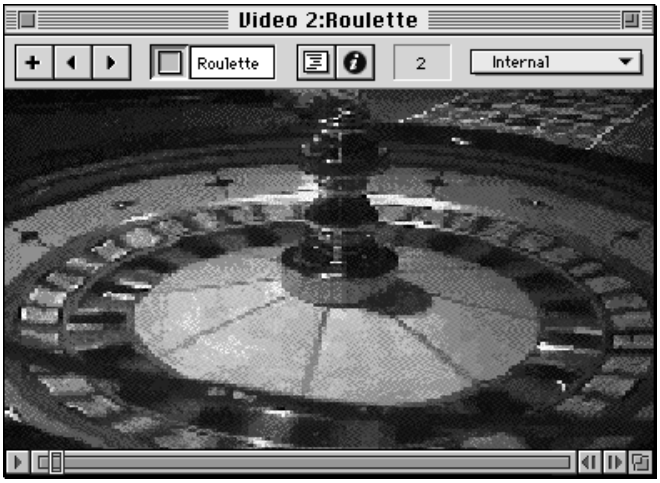


Figure 16-1: Digital Video playback window

The DV window contains the standard buttons common to all media editing windows (see Figure 2-1). It also includes the standard QT controller, which allows you to test videos and perform simple edits on QT cast members. The controller includes a volume slider if the QT movie has an audio track. The DV window does not display a standard controller for AVI cast members, which automatically play once through when they are viewed in the DV window. The D6 video window shows *#digitalVideo* members only. In D7, the Video window was renamed as the QuickTime window and shows *#quickTimeMedia* members only. D7 for Windows has a separate AVI Video window (accessible via `Window ➤ AVI Video`) for AVI (*#digitalVideo*) members.

Playback in the DV window does not obey the cast member property settings, such as *directToStage*, *crop*, *frameRate*, *center*, or *sound*, so it is not an accurate test of these attributes. Test these by placing a DV sprite on the Stage and running Director.

In D6, only QT2-style cast members imported via `File ➤ Import` appear in the DV window; QT3 Assets inserted via `Insert ➤ Media Element ➤ QuickTime 3` include a preview in the QuickTime Xtra Properties dialog box only. In D7, the DV window supports QT3 members.

Table 16-5 summarizes digital video-related commands in the user interface including importing and exporting.

Table 16-5: Digital Video Interface Options

Action	Command
Edit or play a *#digitalVideo* member (D6) or a *#quickTimeMedia* member (D7) in the Video window	Choose Window ➤ Video.[1] Command-9 (Mac). Ctrl-9 (Windows). Double-click on a DV or QT3 cast member or sprite.
View or edit a *#digitalVideo* member's properties (D6) or a *#quickTimeMedia* member (D7)	Use the "i" button in the DV editing window or the Cast window. See Tables 2-8 and 4-8. Modify ➤ Cast Member ➤ Properties
View or edit a *#quickTimeMedia* member's properties in D6.5	Double-click the QT3 thumbnail in the Cast window. Click the *Options* button in the Xtra Cast Member Properties dialog box. Edit ➤ Edit Cast Member
Find *#digitalVideo* cast members (D6 and D7)	Edit ➤ Find ➤ Cast Member ➤ Type: Digital Video
Find *#quickTimeMedia* (QT3) cast members	Edit ➤ Find ➤ Cast Member ➤ Type: Xtra (includes all Xtra types in D6.5 and D7). Edit ➤ Find ➤ Cast Member ➤ Type: QuickTime 3 (D7).
Import *#digitalVideo* (QT2 or AVI) cast members (D6)	Drag and drop from desktop. File ➤ Import: ➤ *Show: QuickTime* (Mac) or *Files of type: Video Clip* (Windows).
Insert *#quickTimeMedia* (QT3) cast members	In D7, drag and drop from desktop. File ➤ Import ➤ QuickTime. Insert ➤ Media Element ➤ QuickTime 3 (D6.5 and D7) In D7, but not in D6.5, click the "i" button, then the Options... button, before clicking the *Browse* or *Internet* button or enter a fileName by hand.
Export QuickTime or AVI movies	File ➤ Export[2,3] *Format: QuickTime Movie* (Mac: D6, D7; Windows: D7 only) *Format: Video for Windows* (Windows only) Xtras ➤ QuickTime Sprite Export[4]

[1] This option is dimmed if QuickTime and AVI are not installed, depending on the version of Director and the platform.
[2] QuickTime export *Options* include setting the frame rate, codec, and size. Transitions will be lost. Director may create multiple audio tracks in the QT movie even when exporting only one sound channel. Add a silent audio track that runs the length of the entire QT movie (to fill any gaps between audio tracks) and use SoundEdit 16 to combine all the sound tracks into one.
[3] QT3 Export requires D7 and the QT3 Export Xtra under both Mac and Windows.
[4] Exports QT3 *#sprite* tracks and requires Apple QuickTime Sprite Export Xtra (*http://www.apple.com/quicktime/developers/tools.html*).

NTSC output

NTSC output resolution is ostensibly 640×480, but the usable screen area is smaller. Leave plenty of border space, avoid one-pixel horizontal lines, and avoid over-saturated colors when creating content for NTSC output.

Prepackaged NTSC filters and NTSC palettes may not give an accurate color. Tweak your colors in Photoshop as you interactively watch the output using a video card that supports an NTSC monitor. Although you can transform graphics to an NTSC palette using `Modify ➤ Transform Bitmap`, Photoshop's `Filter` menu includes video options to de-interlace images and transform them to NTSC-safe colors.

Editing in the Digital Video window

Director's video window can be used for minor edits. Don't expect to use Director for serious DV editing—use an external application instead (see "Applications and Tools" later in this chapter). In D6, you can edit QT2 members (but not QT3 members) on the Macintosh, but cannot edit QTW or AVI cast members under Windows. In D7, you can edit QT3 members both on the Macintosh and under Windows.

You can select a range of frames by `Shift`-clicking in the QT's controller bar in the Video window or by `Shift`-dragging the sliding shuttle. Use the standard `Edit` menu commands to cut, copy, and paste video frames (and the accompanying audio). The QT cast member's duration will update to reflect the video's new length.

Editing a QT cast member inside Director will also modify the original external video file to which it is linked. Make a backup and work on a separate copy of the video instead. If you make a mistake and use `File ➤ Revert`, you'll lose your other changes since your Director file was last saved.

Digital Video Cast Member Properties Dialog Box

The *QuickTime Xtra Properties* dialog box (shown in Figure 16-2) sets QT3 member properties in D6.5 and D7. The *Digital Video Cast Member Properties* dialog box (not shown) is nearly identical, and sets QT2 and AVI member properties.

If multiple DV cast members are selected, you will see only a dialog box with summary information about the multiple cast members (editable properties are settable via Lingo, as indicated in Table 16-8).

Digital video file information

The *Digital Video* and *QuickTime Xtra Properties* dialog boxes include information about the external video file. The first includes a thumbnail, and the latter includes a full video preview. Information includes:

Filename
: The external filename for QT2 members in D6, and QT3 members in D7, is set when importing and will adjust automatically as long as the external video file remains in the same position relative to the Director movie or external castLib. For QT3 members, you may need to click the *Browse* or *Internet* buttons or enter a filename by hand. In D6.5, substitute `@` for the current folder in a QT3 cast member's file path to make it a relative, platform-independent path.

Figure 16-2: QuickTime 3 member properties dialog box

Duration

The duration of the external DV file is listed in seconds. Note that *the duration of member* is measured in units that depend on *the digitalVideoTimeScale* (usually ticks), not seconds.

Dimensions

The width and height should each be a multiple of eight for optimal performance. Refer to *the width of member* and *height of member* properties.

Size

The size listed in the *QT Xtra Properties* dialog box is the true size of the external file. The size listed in the *DV Properties* dialog box is the size of the cast member's data structure and is not related to the size of the external digital video file (see Example 4-6).

The *QT Xtra Properties* dialog box also lists *the timeScale of member* property (usually 600) and the number of tracks.

Playback options

The playback options control whether the audio and video portion of a DV movie play, whether it starts playing automatically, and whether it loops when done:

Video

Controls whether the video tracks(s) of the DV movie are shown. Uncheck this option to improve performance for audio-only DV files.

Sound

Controls whether all audio tracks(s) of the DV movie are audible. In D7, audio-only QT3 members must be played direct-to-Stage. Uncheck this option to improve performance and to prevent sound conflicts with non-QT sounds under Windows for DV movies without audio tracks. To disable an individual audio track, use the *setTrackEnabled* command.

Paused

Controls whether the DV movie plays immediately when it appears in the Score. Check this option to set a DV sprite's attributes or preload the media while it is off-Stage, before bringing it onto the Stage in a subsequent frame. If *Paused* is checked, you must either start the DV sprite by setting the *movieRate of sprite* to 1, or provide the user with a video controller. If you check the *Show Controller* option, the QT movie is paused at the start by default, because Director assumes that the user will control the movie.

Loop

Controls whether the DV file repeats from the beginning or stops after playing once through. If *Loop* is unchecked, *the movieRate of sprite* automatically returns to 0 when a QT movie ends (doesn't work for AVI movies).

To wait indefinitely for a looping video, use the Tempo Channel's *Wait for Cue Point:{Next}* option. If using *Wait for Cue Point:{End}*, the playback head will advance in the Score after the video plays once.

Framing options (cropping and scaling)

The framing options determine the appearance of the digital video cast member within the sprite's bounding box on the Stage:

Crop

Crops the video if the sprite box is smaller than the DV cast member's dimensions. If the sprite box is larger than the cast member, the video is not stretched and the excess area is filled with gray.

Scale

Stretches or shrinks the DV cast member to match the sprite's dimensions. DV sprites will perform better if you stretch/shrink them only in even increments that are multiples of 1/2 or 2 (i.e., 25%, 50%, 100%, or 200%, but not 67%, 125%, or 150%).

Center

Determines whether the video is centered or aligned in the upper-left corner of the sprite box when it is cropped.

QT2 and QT3 assets treat scaling somewhat differently, as shown in Table 16-6. QT3 supports *the scale of member* property that ironically is ignored when the Framing is set to *Scale* mode (*the crop of member* is `FALSE`). The scale of member is set as a list `[xScale, yScale]` where each dimension defaults to 100.0 (normal size), and ranges from 0 (zoom out) to over 50000 (zoom in); 50 is half-size. See also *the translation of member* property.

Table 16-6: QT2 and QT3 Framing Options

Framing	Center	Display	QT2	QT3
Crop[1]	TRUE	Center of video displayed; edges may be cropped.	Cropped, not scaled.	Scaled by *scale of member*, then cropped.
Crop[1]	FALSE	Upper left of video displayed; lower right may be cropped.	Cropped, not scaled.	Scaled by *scale of member*, then cropped.
Scale[2]	N/A	Video stretched or shrunk to match sprite's rect (no cropping).	Scaled based on size of sprite. No cropping.	Same as QT2. *The scale of member* is ignored.

[1] *The crop of member* is TRUE.
[2] *The crop of member* is FALSE.

Stage display options

Table 16-7 summarizes the supported modes for direct-to-Stage video and the DV Controller.

Table 16-7: Support for Direct-to-Stage Video

Platform	DV Format	Direct-to-Stage Setting	Controller	Ink Effects
Macintosh	QuickTime 2	True	Optional	Copy only
Macintosh	QuickTime 2	False	No	Limited[1]
Windows	QuickTime 2	Always true	Optional	Copy only
Windows	QuickTime 3	True	Optional	Copy or custom mask only
Windows	QuickTime 3	False	No	Limited[1]
Windows	VFW (AVI)	True	No	Copy only
Windows	VFW (AVI)	False	No	Limited[1]

[1] Blend is not supported, nor are other processor-intensive inks.

The following options determine whether DV playback bypasses Director's compositing buffer and whether the QT controller is visible.

Direct To Stage

Determines whether the video is played directly to the Stage, or whether it is first passed through Director's offscreen compositing buffer. Direct-to-Stage sprites are always displayed using the Copy ink in the foremost paint layer and always leave trails. See "Drawing to the Stage" in Chapter 1.

Use non-direct-to-Stage playback, which may be very slow, only when you absolutely must overlay other sprites atop, or apply non-Copy ink effects to, a DV sprite. You are ordinarily better off re-designing your product instead.

Director will not automatically refresh the DV sprite area, even after the video terminates. Force Director to redraw the Stage by using a different cast member (a *cover* sprite) in a subsequent frame in the same sprite channel, a fast transition, or using *set the stageColor = the stageColor*.

Show Controller

Determines whether the standard QT controller is shown below a QT sprite, enabling the user to control video playback. This option is available only for QT cast members played direct-to-Stage. For non-direct-to-Stage QT or AVI files, which do not support controllers, you must provide a custom controller via Lingo. The standard QT controller includes a sound volume control only if the QT file contains an audio track. D7 provides a custom QuickTime Control Slider Behavior under `Window ➤ Library Palette ➤ Media ➤ QuickTime`.

The standard controller appears in the foremost paint layer, as does the entire video sprite when played direct-to-Stage. The controller itself is drawn using the colors from positions 1, 7, 248, and 256 of the palette. If you are using a custom palette under Windows, reserve the first and last ten colors to be the same as those in the Windows System palette to avoid conflicts.

When *Show Controller* is checked, the QT sprite is automatically paused at first, and will not play until started by setting *the movieRate of sprite* to *1* or via the controller. See the *Paused* option and *the pausedAtStart of member* property.

Video rate

The following options determine the speed and synchronization mode of DV playback. The frame rate at which a DV sprite plays is completely independent of the Score's Tempo channel, *the frameTempo*, and *the puppetTempo*.

Video

Controls the playback synchronization mode of the DV movie, and has two possible settings:

Sync to Soundtrack

Plays the DV file in the customary time-based manner. The QuickTime and Video for Windows playback engines drop video frames to maintain audio synchronization if necessary. The playback rate is determined by the DV movie's intrinsic frame rate, which is defined when the video is created in your DV editing software.

Play Every Frame (No Sound)

Plays the DV file as if it were a straight animation, and QT or VFW does not drop frames to maintain synchronization. This mode disables the audio track(s), and is most appropriate for visual transitions in which you want to see every frame of the animation. The apparent frame rate is ordinarily slower than in the *Sync to Soundtrack* mode, as frames are never dropped.

Rate

Determines how fast Director attempts to play the video in *Play Every Frame (No Sound)* mode (ignored in *Sync to Soundtrack* mode). Video frames will not be dropped if the target rate cannot be met, so the actual playback rate might be slower.

Normal

Plays every frame of the DV file, but no faster than its intrinsic frame rate.

Maximum

Plays every frame of the DV file as fast as it can.

Fixed

Plays every frame of the DV file at the rate specified in the adjacent *Frames Per Second* field.

Frames Per Second

Specifies the rate for *Fixed* playback of the DV file when using *Play Every Frame (No Sound)* mode. The default value is the DV file's intrinsic frame rate, which is defined when the video is created in your DV editing software.

Memory management options

The *Enable Preload* and *Unload* options control how the DV cast members are loaded and unloaded from memory.

The Lingo equivalent for each of the options in the *Digital Video and QuickTime Xtra Properties* dialog box are shown in Table 16-8.

Table 16-8: Lingo Equivalents to DV and QT Xtra Property Dialog Box

Dialog Option	**Lingo Equivalent**	**See Also**
Cast Member Name	the name of member	castLibNum of member
File Name	the fileName of member (includes path)	name of member
Dimensions	the width, height, and rect of member	width, height, and rect of sprite
Duration	the duration of member / float (the digitalVideoTimeScale)	movieTime of sprite
Size	Size of header for QT2 and external file size for QT3. See Example 4-6.	size of member, duration of member
FrameRate/ Frames Per Second	the frameRate of member. See "Determining a DV's intrinsic frameRate."	movieRate of sprite
Video	the video of member[1]	sound of member
Sound	the sound of member[2]	video of member, volume or volumeLevel of sprite, soundLevel, soundEnabled, trackEnabled, setTrackEnabled

Table 16-8: Lingo Equivalents to DV and QT Xtra Property Dialog Box (continued)

Dialog Option	**Lingo Equivalent**	**See Also**
Paused	the pausedAtStart of member (unrelated to the *pause* command)	controller of member, movieRate of sprite
Loop	the loop of member	movieTime of sprite, duration of member
Crop	the crop of member = TRUE	center of member, scale of member
Scale	the crop of member = FALSE	center of member, width, height, stretch, and rect of sprite
Center	the center of member	crop of member
Direct to Stage	the directToStage of member	ink, controller, invertMask, and mask of member
Controller	the controller of member	directToStage, pausedAtStart of member
Sync to Soundtrack	the frameRate of member = 0	movieRate of sprite
Play Every Frame (No Sound)		
Normal	the frameRate of member = –1	movieRate of sprite
Maximum	the frameRate of member = –2	movieRate of sprite
Fixed fps	the frameRate of member = *n* (*n* > 1, specified in fps field)	fps field in dialog box
Enable Preload	the preLoad of member	the preLoadRAM, preLoad-Member
Unload Settings (Normal/ Next/Last/Never)	the purgePriority of member = *n* 0 <= *n* <= 3	unLoad, unLoadMember

[1] Changing *the video of member* also resets *the movieTime of sprite* to 0, which restarts the movie at the beginning.
[2] Setting *the sound of member* to FALSE initializes *the volume of sprite* to 0. Changing *the volume of sprite* via Lingo or the QT controller overrides *the sound of member* setting.

Digital Video Properties and Functions

Table 16-9 lists the properties for both *#digitalVideo* (QT2 and AVI) and *#quick-TimeMedia* (QT3) cast members, including those that are new or behave differently with *#quickTimeMedia* cast members than with *#digitalVideo* cast members. Consult Table 4-10 for a list of other sprite and member properties that are not specific to digital video cast members. See also the HTML documentation that comes with the D6.5 QT3 Asset Xtra and the D7 online Help.

In Table 16-9, with the exception of *the digitalVideoTimeScale*, which is a system property, it is implied that each property takes the form:

```
the property of member qtMember
```

and/or:

```
the property of sprite qtSprite
```

For example, *the movieRate* is a sprite property and can be set using:

```
set the movieRate of sprite 5 = 1
```

or in D7 syntax:

```
sprite(5).movieRate = 1
```

Note that sprite properties override any member property of the same name on a per-sprite basis. The default value, if any, is shown in italic in the *Value* column. QT3 imports many media types, but not all commands and properties are applicable for all media types (see Table 16-10).

To create a QT3 cast member in D6.5, choose `Insert` ➤ `Media Element` ➤ `QuickTime 3`. Use `File` ➤ `Import` to create QT3 members.

Table 16-9: Digital Video and QuickTime 3 Properties

Command	Value	Read-Only	Member	Sprite	Notes
bottom	Pixels from top of Stage.			✓	Includes height of controller, if any.
center	TRUE | *FALSE*		✓		Used only if *crop* = TRUE.
controller	TRUE | *FALSE*		✓		Used only if *directToStage* = TRUE.
crop	TRUE | *FALSE*		✓		See *center, scale, translation* entries. If *crop* is FALSE, movie is scaled to sprite box.
digitalVideoTime Scale	Default is 60 units/second.				A system property, not a sprite or member property.
digitalVideoType	#quickTime | #videoFor-Windows	✓	✓		Always *#quickTime* for QT3 members.
directToStage[1]	*TRUE* | FALSE		✓		See *controller, mask, mouseLevel* entries.
duration	Measured in ticks.	✓	✓	✓	See *mTime, movieTime* entries.
fileName[2,3]	Path to external video.		✓		Replace folder name in file path with @ in D6.5.
frameRate of member	0 | -1 | -2 | *n*		✓		See *mRate and movieRate* entries.

Table 16-9: Digital Video and QuickTime 3 Properties (continued)

Command	Value	Read-Only	Member	Sprite	Notes
height	Measured in pixels.		✓	✓	Should be multiple of 8.
ink	Default is 0 (Copy).			✓	Copy ink is used for direct-to-Stage sprites.
invertMask[4]	TRUE | *FALSE*		✓		See *mask* entry.
isVRmovie[4]	TRUE | FALSE	✓	✓	✓	See Table 16-10.
left	Pixels from left of Stage.			✓	Should be multiple of 8.
loaded	TRUE | FALSE		✓		Indicates only whether header is loaded, not data.
loop	TRUE | *FALSE*		✓		See *loopBounds* entry.
loopBounds[4]	[*startTicks*, *endTicks*]			✓	[0, 0] indicates no *loopBounds*.
mask[4]	member *oneBitMember*		✓		Set to 1-bit cast member or zero to disable; Only if *directToStage* = TRUE.
media	Not useful in most cases.		✓		Does not access external DV file data.
modified	TRUE | FALSE		✓		Indicates whether video has been edited in Video window.
mouseLevel[4,5]	#none | *#controller* | #all | #share			✓	See *VRTriggerCallback* in Table 16-10.
movieRate[6]	0 – n, default is 1.0.			✓	Use *mRate* instead for QT3 sprites in SW6.0.1 and D6.5.
movieTime	Measured in ticks by default.			✓	Use *mTime* instead for QT3 sprites in SW6.0.1 and D6.5.
mRate[4,6]	0 – n, default is 1.0.			✓	See *pausedAtStart*; use for SW6.0.1 and D6.5 instead of *movieRate.*
mTime[4]	Measured in ticks.			✓	Use for SW6.0.1 and D6.5 instead of *movieTime*.
name	String.		✓		Defaults to initial external file name, excluding the path.
purgePriority	0 | 1 | 2 | *3* (default is 3: Normal)		✓		Indicates whether header is purged. Streamed media is always purged.

Table 16-9: Digital Video and QuickTime 3 Properties (continued)

Command	Value	Read-Only	Member	Sprite	Notes
rect	Measured in pixels.		✓	✓	Dimensions should be multiple of 8, and sprite rect should be even multiple of member rect.
regPoint	point (*x, y*)		✓		Defaults to center.
right	Pixels from left of Stage.			✓	Should be multiple of 8.
rotation[4]	0.0 to 360.0 degrees.		✓	✓	Default is 0.
scale[4]	[*xScale, yScale*]		✓	✓	Used if *crop* = TRUE. Default is [100.0, 100.0].
size[2]	Bytes.				Size of header for QT2. Size of external file for QT3.
sound	*TRUE* | FALSE		✓		Ignored if *frameRate* is not 0 (sync to soundtrack).
startTime	Ticks.			✓	Unreliable.
stopTime	Ticks.			✓	Unreliable.
timeScale	Units/second.	✓	✓	✓	Defaults to 600 for QT on Mac. See *the digitalVideoTimeScale* entry.
top	Pixels from top of Stage.			✓	Should be multiple of 8.
trails	TRUE | FALSE			✓	If *directToStage* = TRUE, DV leaves trails.
translation[4]	[*hPixels, vPixels*]		✓	✓	Used if *crop* = TRUE.
type[2]	*#digitalVideo* (QT2 or AVI) | *#quickTimeMedia* (QT3)	✓	✓		See *digitalVideoType* entry.
video	*TRUE* | FALSE		✓		See *sound* entry.
volume	0 to 255 (or higher)[7]			✓	Always reports 0 for QT3 in D6.5; see *volumeLevel* entry.
volumeLevel[4]	0 to 255 (or higher)[7]			✓	Use in D6.5 instead of *volume*.
width	Measured in pixels.			✓	Should be multiple of 8.

[1] If *the directToStage of member* is FALSE, the user cannot interact with QTVR movies automatically. You must manually simulate callbacks with *VRPtToHotSpotID( )*.
[2] Different for QT3 cast members and sprites than for QT2 cast members and sprites.
[3] The default *fileName* for QT3 cast members in D6.5 is an absolute path. In the *QuickTime Xtra Properties* dialog box, replace the current folder in the path with the "@" operator to create a relative path (not necessary in D7).
[4] New as of D6.5. Applies to QT3 cast members and sprites, not QT2 members and sprites.
[5] If *the mouseLevel of sprite* is *#none*, callbacks are not sent. You must use *VRPtToHotSpotID( )* to manually handle the callbacks. The docs are ambiguous, the correct syntax is *VRPtToHotSpotID(sprite n, point(the mouseH, the mouseV))*.
[6] The *mRate* and *movieRate* return 0 if *the frameRate of member* is not 0 (*Sync to Soundtrack* mode).
[7] Setting *the volumeLevel* or *volume of sprite* higher than 255 causes distortion.

Table 16-10 lists the sprite properties that pertain only to QTVR assets imported as *#quickTimeMedia* members. All are settable except *the VRNodeType*. For example:

```
set the VRmotionQuality of sprite 5 = #minQuality
```

The VR prefix for each property is unnecessary in D7. For example:

```
sprite(5).motionQuality = #minQuality
```

Table 16-10: QTVR-Specific Properties of QT3 Assets

Command	**Value**	**Notes**
VRFieldOfView	*degrees*	Current field of view.

VRHotSpotEnterCallback	#enterHotspot | 0	`on enterHotspot me, hotSpotID`
		`  actions`
		`end`

VRHotSpotExitCallback	#exitHotspot | 0	`on exitHotspot me, hotSpotID`
		`  actions`
		`end`

VRMotionQuality	#minQuality | #maxQuality | #normalQuality	See VRStaticQuality entry.

VRMovedCallback	#movedVR | 0	`on movedVR me`
		`  actions`
		`end`

VRNode	nodeID	Current node ID being displayed.

VRNodeEnterCallback	#nodeEnter | 0	`on nodeEnter me, nodeID`
		`  actions`
		`end`

VRNodeExitCallback	#nodeExit | 0	`on nodeExit me, oldNode, newNode`
		`  actions`
		`  return 0 | 1`
		`end`
		0: cancel (not implemented)
		1: allow (go to new node)

VRNodeType	#panorama | #object | #unknown	#unknown means non-QTVR; see isVRMovie entry in Table 16-9.
VRPan	*degrees*	See VRTilt entry.
VRStaticQuality	#minQuality | #maxQuality | #normalQuality	See VRMotionQuality entry.
VRTilt	*degrees*	See VRPan entry.

Table 16-10: QTVR-Specific Properties of QT3 Assets (continued)

Command	Value	Notes

VRTriggerCallback	#triggerVR | 0	MouseLevel must be #all or #share, not #none or #controller.
		`on trigger VR me, hotSpotID`
		`  actions`
		`  return 0 | 1`
		`end`
		0: cancel
		1: continue

VRWarpMode	#none | #partial |#full	See VRStaticQuality, VRMotion Quality entries.

Table 16-11 lists the supported QuickTime 3 functions. See the HTML help files included with Director 6.5 under the *Help 6_5/QT3 Help* folder, or the D7 online Help. QT3 also supports the track sampling and time functions in Tables 16-15 and 16-16.

Table 16-11: QuickTime 3 Asset Xtra Commands and Functions

Command	System	Member	Sprite
canUseQuicktimeStreaming()[1]	✓		
isUsingQuicktimeStreaming()[1]	✓		
interface (xtra "QuickTimeSupport")	✓		
mMessageList (xtra "QuickTimeSupport")	✓		
new (#quickTimeMedia)	✓		
qtRegisterAccessKey (*category, key*)	✓		
qtUnRegisterAccessKey (*category, key*)	✓		
quickTimeVersion()[2]	✓		
setTrackEnabled (sprite *qtSprite, track*)			✓
trackCount (member *qtMember*) trackCount (sprite *qtSprite*)		✓	✓
trackEnabled (sprite *qtSprite, track*)			✓
trackText (sprite *qtSprite*, *track*)			✓
trackType (member *qtMember, track*) trackType (sprite *qtSprite, track*)		✓	✓

Table 16-11: QuickTime 3 Asset Xtra Commands and Functions (continued)

Command	System	Member	Sprite
useQuickTimeStreaming (TRUE | FALSE)[1]	✓		
VREnableHotSpot (sprite *qtvrSprite*, *hotSpotID*, TRUE | FALSE)[3]			✓
VRGetHotSpotRect (sprite *qtvrSprite*, *hotSpotID*)[3]			✓
VRNudge (sprite qtvrSprite*qtvrSprite*, #left | #upLeft | #up | #upRight | #right | #downRight | #down | #downLeft)[3]			✓
VRPtToHotSpotID(sprite *qtvrSprite*, point(the mouseH, the mouseV))[3]			✓
VRswing (sprite *qtvrSprite*, *pan*, *tilt*, *fieldOfView*)[3,4]			✓

[1] Streaming QT is "not implemented and extremely unsupported" in D6.5 and D7, according to Macromedia. For streaming video, see RealVideo from Real Networks (formerly Progressive Networks) at *http://www.real.com*.
[2] See Table 16-2.
[3] The "VR" prefix is deprecated in D7. Use *enableHotSpot( ), getHotSpotRect( ), nudge( ), ptToHotSpotID( ),* and *swing( )* in D7.
[4] The *pan*, *tilt*, and *fieldOfView* are in degrees.

Masks

QT3 adds support for 1-bit cast member masks (see *the mask of member*), which allow non-rectangular QT movies to play direct-to-Stage. If *the invertMask of member* is `FALSE`, black areas of the mask show the QT cast member, and white areas hide it. If *the invertMask of member* is `TRUE`, the reverse holds.

Don't use a duplicated member as a mask. (Duplicating a member doesn't always generate a unique internal ID, which the QT3 Xtra uses to track the mask.) Use copy and paste to create a new cast member with a unique internal ID.

The registration point of the mask member is placed at the upper-left corner of the QT3 sprite. This allows you to use a shape cast member as a mask. If using a bitmap, set the mask's registration point to its upper-left corner.

Controlling Digital Video Playback

The *movieRate of sprite* and *frameRate of member* control the playback rate of DV sprites. In D6.5, QT3 cast members use *the mRate of sprite* instead of *the movieRate of sprite*. In all other versions, including D7, use *the movieRate* property instead of *the mRate*.

Synchronous Video Playback

Digital videos are usually played synchronously, using *Sync to Soundtrack* mode, in which audio and video are synchronized to a time code, and QuickTime (QT) or Video for Windows (VFW) will drop frames if necessary to achieve the requested frame rate. The *movieTime of sprite* is used to start and stop the video and even play it backward.

If the natural frame rate of the movie is 10 fps, setting the *movieRate* to 1 will play the movie at 10 fps; setting the *movieRate* to 2 will play the movie at 20 fps, and

so on. Table 16-12 summarizes the different playback settings. For the purposes of Table 16-12, *the frameRate of member* should be zero (0), and *the movieRate of sprite* should be set to the value *n* shown in the table.

Table 16-12: Synchronous Playback Speeds

DV Speed	movieRate or mRate	Notes
Stopped	*n* = 0	See *pausedAtStart* and *controller of member* in Table 16-9.
Normal speed	*n* = 1	DV plays at intrinsic frame rate defined when it was created.
Reverse normal speed	*n* = –1	Backward video playback is inefficient and may be jerky. Sounds play in reverse.[1]
Fast forward	*n* > 1	Sound plays quickly and at a higher pitch. Video may be jerky at high multiples.
Fast reverse	*n* < –1	Sounds play quickly, at higher pitch, and in reverse. Video will be jerky.
Slow-motion forward	0 < *n* < 1.0	Sounds play slowly and at a lower pitch.
Slow-motion reverse	–1.0 < *n* < 0	Video may be jerky. Sounds play slowly, and at a lower pitch.[1]

[1] Sounds play in reverse and occasionally sound like the word "Satan."

Asynchronous Video Playback

To play back every frame of a DV file without regard to the time code, use the *Play Every Frame (No Sound)* option. This is commonly used for animations or visual transitions when you don't want QT or VFW to drop frames, and don't care about speed.

Table 16-13 summarizes the possible settings for asynchronous video playback using *the frameRate of member*. Setting *the frameRate of member* at runtime is not reliable; use the *DV Cast Member Property* dialog box to set it instead. Set *the movieRate of sprite* to 1 to play the video forward, to 0 to pause the video, and to –1 to play the video backward.

Table 16-13: Asynchronous Playback Speeds

Playback Speed	frameRate	Video	Audio
Synchronized	f = 0	Drops frames, if needed, to achieve intrinsic rate, multiplied by *movieRate of sprite*. See Table 16-12.	Yes[1]
Normal	f = –1	Played at the DV file's intrinsic frame rate without dropping frames.	None
Maximum	f = –2	Played as fast as possible without dropping frames.	None
Fixed	1 <= f <= 255	Played at the specified frame rate without dropping frames.	None

[1] Audio is synched to the video, if audio track is enabled. See *the sound of member* property and *setTrackEnabled( )* function.

Determining a DV's intrinsic frameRate

Example 16-4 determines the frame rate that was assigned to a QT2 (*#digitalVideo*) video file when it was created in Adobe Premiere, for example. It makes use of a trick that doesn't work with QT3 (*#quickTimeMedia*) members. For QT3 members, examine the frame rate in MoviePlayer.

Example 16-4: Reading a DV File's Intrinsic Frame Rate

```
on getFrameRate DVmember
  set oldRate = the frameRate of member DVmember
  -- Set an invalid frame rate and it will revert to the true rate.
  set the frameRate of member DVmember = -3
  set intrinsicRate = the frameRate of member DVmember
  set the frameRate of member DVmember = oldRate
  if intrinsicRate > 0 then
    return intrinsicRate
  else
    return 0
  endif
end getFrameRate
```

Playing Digital Video from/to a Specific Point

The *startTime of sprite* (which defaults to zero) and *stopTime of sprite* (which defaults to the duration of the DV cast member) properties can be used to cue specific segments of a digital video file. I have found them to be unreliable (they work best if set once the video is playing). Instead, use *the movieTime of sprite* property to set or read the position in a DV file.

The movieTime of sprite property can be set when the video is playing or stopped, as indicated by *the movieRate of sprite*. Table 16-14 assumes that *the digitalVideoTimeScale* is set to 60, its default. Note that D6.5 QT3 members use the *mRate* and *mTime of sprite* properties instead of the *movieRate* and *movieTime*. The *movieTime of sprite* is affected by the *digitalVideoTimeScale*, but the *mTime of sprite* is not.

Table 16-14: Digital Video Timing

Position Within DV	**movieTime or mTime of sprite Equivalent**
Beginning of video	0
t ticks from the beginning	*t*
s seconds from the beginning	*s* * 60
Current time of DV (usually in ticks)	the movieTime of sprite *dvSprite* the mTime of sprite *dvSprite*
Current time of DV in *milliseconds*	the currentTime of sprite *dvSprite*
Current time – *s* seconds	(the movieTime of sprite *dvSprite*) – *s* * 60 (the mTime of sprite *dvSprite*) – *s* * 60

Table 16-14: Digital Video Timing (continued)

Position Within DV	movieTime or mTime of sprite Equivalent
Current time + *s* seconds	(the movieTime of sprite *dvSprite*) + *s* * 60 (the mTime of sprite *dvSprite*) + *s* * 60
Midpoint of DV	0.5 * (the duration of the member of sprite *dvSprite*)
s seconds from the end of video	(the duration of the member of sprite *dvSprite*) – *s* * 60
t ticks from the end of video	(the duration of the member of sprite *dvSprite*) – *t*
End of video	the duration of the member of sprite *dvSprite*
*n*th Photo – JPEG still frame	*n* *60 / (video frame rate in fps[1])
Previous keyframe	trackPreviousKeyTime(sprite *dvSprite*, *whichTrack*)
Next keyframe	trackNextKeyTime(sprite *dvSprite*, *whichTrack*)

[1] See Example 16-4.

Analyzing and Controlling Individual Digital Video Tracks

A DV file can contain multiple tracks—often one video track and one audio track. There can be 0 to 7 video or audio tracks, plus other media types, including text, MIDI, and sprites (unrelated to Director's sprites).

Lingo provides track-specific commands, summarized in Table 16-15. Although Lingo may report that a track exists and is enabled, this may not reliably reflect whether it will actually be played. QuickTime for Windows Version 2.1.2 and earlier support only one audio track and one video track and do not support text tracks. AVI files are limited to one audio and one video track.

Table 16-15: Track-Specific Digital Video Commands

Action	Command	Notes
Count the number of tracks in a DV sprite or cast member	trackCount (sprite *y*) trackCount (member *x*)	See trackType( )
Determine the type of data in a track	trackType (sprite *y*, *t*) trackType (member *x*, *t*)	Returns #video, #sound, #text, #music, #sprite[1], #unknown, #chapter, or VOID if out of range.
Retrieve the text content of a text track	trackText (sprite *y*, *t*)	Applies only to text tracks.
Determine whether a track is enabled in a DV sprite[2]	trackEnabled (sprite *y*, *t*)	See setTrackEnabled( ), the video, and the sound of member.
Enable/disable a track in a DV sprite	setTrackEnabled (sprite *y*, *t*) = TRUE | FALSE	See trackEnabled( ), the video, and the sound of member.

Table 16-15: Track-Specific Digital Video Commands (continued)

Action	Command	Notes
Determine when a track starts within a DV sprite or cast member[3]	trackStartTime (sprite *y*, *t*) trackStartTime (member *x*, *t*)	See the startTime, movieTime, and mTime of sprite.
Determine when a track ends within a DV sprite or cast member[3]	trackStopTime (sprite *y*, *t*) trackStopTime (member *x*, *t*)	See the stopTime, movieTime, and mTime of sprite, the duration of member.
Enable/disable all video tracks in a cast member, and any sprites that reference it	set the video of member *x* = TRUE | FALSE	See setTrackEnabled(), trackEnabled(), sound of member.
Enable/disable all audio tracks in a cast member, and any sprites that reference it	set the sound of member *x* = TRUE | FALSE	See setTrackEnabled(), track Enabled(), video of member.

[1] The #sprite tracks create by the QuickTime Sprite Export Xtra return VOID when imported as a QT2-style #digitalVideo member, but return #sprite when inserted as a QT3-style #quickTimeMedia member.
[2] Tracks are enabled/disabled on a per sprite basis, not a per cast member basis.
[3] The *trackStartTime( )* specifies the offset of a track within the digital video file, which is ordinarily 0 (the beginning of the DV file). The *trackStartTime( )* and *trackStopTime( )* are *not* analogous to *the startTime* and *stopTime of sprite*.

Timing Within Digital Video Tracks

Lingo provides the *trackPreviousKeyTime*, *trackNextKeyTime*, *trackPreviousSampleTime*, and *trackNextSampleTime* functions to access specific data points within individual tracks as shown in Table 16-16. These functions do not appear to return reliable data. Lingo also allows you to query the time scale being used to play digital video files. See also Chapter 11, *Timers and Dates*, in *Lingo in a Nutshell*.

Table 16-16: Track Sampling and Time Functions

Action	Command
Determine time of next keyframe	set nextKey = trackNextKeyTime(sprite *y*, *trackNum*)
Determine time of previous keyframe	set prevKey = trackPreviousKeyTime(sprite *y*, *trackNum*)
Determine time of next sample	set nextSample = trackNextSampleTime(sprite *y*, *trackNum*)
Determine time of previous sample	set prevSample = trackPreviousSampleTime(sprite *y*, *trackNum*)
Determine or set the time units used to measure time for digital video cast members	the digitalVideoTimeScale, where scale = 1/t (i.e., 60 = 1/60th of a second)
Determine or set the time units on which the digital video's frames are based	the timeScale of member, where scale = 1/t (i.e., 600 = 1/60th of a second)

Common Digital Video Operations

Although you can wait for a video via the Tempo channel, here are some Lingo variations. Example 16-5 waits for a digital video by comparing the current time to

the overall length of the video. Waiting in a *repeat* loop devotes all the time to video playback, but locks out other interactivity.

Example 16-5: Waiting for a Video in a Repeat Loop

```
on waitForVideo channel
  set end = the duration of the member of sprite channel
  repeat while the movieTime of sprite channel < end
    updateStage
  end repeat
end
```

Example 16-6 waits until a digital video stops by itself or until the user clicks the mouse. *The movieRate of sprite* automatically becomes 0 when a QT sprite ends, but when using AVI sprites, the playback head won't advance until the user clicks.

Example 16-6: Allowing a Mouse Click to Interrupt a Video

```
on exitFrame
  global gVideoChan
  if the movieRate of sprite gVideoChan then
    if the mouseDown then
      set the movieRate of sprite gVideoChan = 0
    end if
    go the frame
  end if
end
```

Table 16-17 lists common digital video operations that can be achieved via Lingo. Refer also to Table 16-14. The last two entries in the table can be used to reduce audio conflicts under Windows.

Table 16-17: Common Digital Video Operations

Action	Command
Rewind a video	`set the movieTime of sprite x = 0`
Determine whether a video is currently playing	`if the movieRate of sprite x <> 0 then...`
Determine whether a video file has reached its end	`if the movieTime of sprite x >= the duration of the member of sprite x then...`
Wait until a specific cue point has been reached in the DV file	See the Tempo channel's *Wait for Cue Point* option or *the mostRecentCuePoint of member* and *isPastCuePoint( )*.
Wait until a specific time in seconds is reached in the DV file	`if the movieTime of sprite x ¬ <= t * 60 then go the frame`
Wait until a specific time in seconds is reached in the DV file while in a repeat loop	`repeat while the movieTime of sprite x <= t * 60 updateStage end repeat`

Table 16-17: Common Digital Video Operations (continued)

Action	Command
Shut off all DV-related audio	`set the movieRate of sprite x = 0`

Shut off all non-DV related audio	`repeat with x = 1 to 8`
	`  puppetSound x, 0`
	`  sound stop x`
	`end repeat`

Digital Video Resources

There are many QT-related URLs of interest. Here are a select few. See *http://www.zeusprod.com/nutsbell/links.html* for many more.

Information

Getting started with QuickTime 3:

http://www.apple.com/quicktime/information/index.html

QuickTime Gazette:

http://www.Blackstone.ca/QuickTimeGazette/Latest_News.html

QuickTime FAQ (outdated but worthwhile):

http://www.QuickTimeFAQ.org/

Creating QuickTime content:

http://www.apple.com/quicktime/authors/index.html

QuickTime developer mailing list (free, informal support):

http://lists.apple.com/quicktime-dev.html

QuickTime announce mailing list (news and announcements):

http://lists.apple.com/quicktime-announce.html

Applications and Tools

Here are some of the most common digital video editing, management, compression, and analysis tools. Recall also that Director for the Macintosh exports QT files and Director 7 for Windows exports AVI or QT files. When timing is critical, you can export the Score animation to a digital video file, then reimport the resulting video for playback in Director.

Adobe Premiere creates and edits DV files (Mac/Win):

http://www.adobe.com

SoundEdit 16, included with Director 6 Multimedia Studio for Macintosh manipulates QT audio tracks, and adds cue points (Mac only):

http://www.macromedia.com

QuickTime 3 Pro (upgrades to MoviePlayer and PictureViewer) plays and performs minor edits on DV files, converts between formats, and prepares files for Windows playback (Mac/Win):

http://www.apple.com/quicktime/information/qt3pro.html

deBabelizer by Equilibrium creates custom palettes for video, etc. (Mac/Win):

http://www.equilibrium.com

MediaCleaner Pro (formerly MovieCleaner) from Terran Interactive—the standard CD compression tool. Not an editing tool:

http://www.terran-int.com

QT Gallery 1.0, Lakewood Software is an image editor and browser tool supporting QT3's image import options (PICT, JPEG, TIFF, GIF, and Photoshop, etc.), editing filters, and transition effects:

http://www.lakewoodsoftware.com

Make Effects movie—adds QT3 transition effects:

http://www.apple.com/quicktime/developers/tools.html
ftp://ftp.apple.com/Quicktime/developers/makeeffectmovie.sea.hqx

Video-Related Xtras

Besides the QT3 Xtra from Macromedia included with D6.5 and D7, see the following Xtras.

MPEG and DirectMedia Xtras—Tabuleiro da Baiana:

http://www.tbaiana.com

MultiMixer Xtra by TurnTable controls the pan of a QuickTime movie's sound:

http://www.turntable.com/multimixer

RobinHood Xtra for QT3:

http://www.hermes.de/beise/robin.html

Focus 3 Xtra—captures and displays video from device connected to a Macintosh:

http://www.focus3.com

VSnap—video capture:

http://www.penworks.com

Digital Video Troubleshooting

When troubleshooting DV, you must narrow down the problem. Does the problem occur in MoviePlayer or only in Director? Does the problem occur under Windows 3.1, Windows 95/98, Windows NT, and the Macintosh? Does it occur with all videos, or only some? Does it depend on whether there are other sounds or other videos playing? Does the problem depend on the version of QuickTime or the sound or video card installed? Are RSX and DirectSound installed under Windows? By answering these questions, you'll reveal the most likely culprit.

Test your video in an external application, such as MoviePlayer, before blaming Director. Test with a known working video—one with one video track, one audio track, and a low data rate—that you can substitute for a suspicious video during testing. Have a known working test machine available.

Lingo errors

The Lingo script errors "Not a digital video sprite," "Digital video sprite expected," or "Property not found" indicate that you are attempting to check or set an invalid member or sprite property.

For example, *the movieTime of sprite* is valid for only digital video sprites. Ensure that you are in the correct frame in which the DV sprite appears and that you are using the correct sprite channel. Check *the type of the member of sprite n* to ensure it is a *#digitalVideo* or *#quickTimeMedia* member.

Many QT3 properties (see Tables 16-9 and 16-10) are not supported for older QT2-style cast members.

Video doesn't appear

Ensure that other videos play and that the proper version of QT, QTW, or VFW is installed (see Examples 16-1 and 16-2 and Tables 16-1 and 16-2). During authoring, Director will display an error message if QuickTime or Video for Windows is not loaded and cast members requiring those extensions are present. Projectors may simply fail to display digital video sprites if the required extensions are missing.

Include the externally linked DV file in the correct relative path.

Test the video in an external video application. Ensure that it contains a video track that is enabled (see the *trackCount()*, *trackType(), trackEnabled()*, and *setTrackEnabled()* functions in Table 16-15). When using QTW2 under Windows, ensure that the video was prepared specifically for Windows playback (i.e., flattened and deforked).

Test the video in Director's Video window. The beginning of many videos is black, so you may not see an image on Stage until they play. Ensure that the DV sprite is located on the Stage, not off-Stage, and make sure that the Director playback head is moving.

Set *the video of member*, *the visible of member*, and *the directToStage of member* to `TRUE`. Set *the puppet of sprite* to `FALSE`. If playing the video non-direct-to-Stage, ensure that the DV sprite is not obscured by another sprite. See "Drawing to the Stage" in Chapter 1 for generic reasons why a sprite may not appear, such as the wrong size, width, or location.

Some versions of QuickTime for Windows and Video for Windows may not support multiple video tracks.

Video doesn't play from a different drive (can't find video)

If Director asks, "Where is movie *xxxx*?" be sure that you've included the external DV files with you Projector. Maintain the same relative path position after protecting your Director files or creating your Projector.

In D6.5, QuickTime 3 cast members will not adjust their paths automatically, as will QT2 cast members, unless you use the `@` operator to specify the path relative to the Director movie's current folder. In D7, this is not an issue.

Cast members inserted via `Insert ➤ Media Element ➤ QuickTime 3` in D6.5 or D7, or imported via `File ➤ Import` in D7, require the appropriate runtime version of the QuickTime 3 Asset Xtra.

Video works on Macintosh but not under Windows

Ensure that the movie was flattened and deforked. (Save it as *Self-Contained* and *Playable on non-Apple computers* using MoviePlayer.)

If you imported the video using `File ➤ Import` in D6.5 or earlier versions, it is treated as a QT2-style *#digitalVideo* member. It will play if either QT2 or QT3 is installed on the Macintosh, but requires QTW2 under Windows. (You must install the 16-bit version of QTW2 when using a 16-bit Projector and the 32-bit version of QTW when using a 32-bit Projector.) QTW2-style *#digitalVideo* members will not work under Windows if only QTW3 is installed.

Similarly, *#quickTimeMedia* members require QT3 and the QT3 Asset Xtra. QT2 or QTW2 will not play *#quickTimeMedia* members.

Any video using any new features of QT3 (see the next section) must be imported using `Insert ➤ Media Element ➤ QuickTime 3` to be supported in D6.5 for Windows. In D7, all QuickTime videos are imported as QT3 members by default.

Sorenson video caveats

The Sorenson Video Codec is much more processor-intensive than Cinepak compression. However, Sorenson-compressed video at 70 KB/sec may look better than Cinepak-compressed video at 320 KB/sec. Therefore, use a lower data rate when using the Sorenson codec.

Sorenson Video is a QT3-specific codec requiring the QT3 Xtra under Windows. If you import a Sorenson-compressed QT movie as a QT2-style cast member using `File ➤ Import` in D6.x, it *will* work on a Macintosh with QT3 installed, but will *never* work under Windows regardless of the QTW version(s) installed. You must use `Insert ➤ Media Element ➤ QuickTime 3` in D6.5 to create a *#quickTime-Media* cast member to make a Sorenson video playable under Windows (assuming QTW3 and the QT3 Asset Xtra are installed). Again, in D7, all videos are imported as QT3 by default.

Video palette problems

If, on the Macintosh, the first QT movie played looks fine, but the second QT movie has the wrong palette, you need the FixPalette XObject. It is necessary only when using multiple QT movies with different custom palettes on the Macintosh. It

is not necessary in D7, under Windows, or when using a single custom palette. See "Using the FixPalette XObject" online at *http://www.zeusprod.com/technote/patchpal.html*.

If your digital video appears to be using the wrong palette under Windows, it is probably a general palette issue, and not specific to DV. There are several fixes that work for both standard graphics and DV, discussed in Chapter 13.

Poor visual appearance

Remember the phrase "Garbage in, garbage out." You can't get high-quality video from a low-quality source (such as standard VHS tape). The video won't look any better in Director than it does in MoviePlayer. If the video looks acceptable in 16-bit (thousands of colors) but not in 8-bit (256 colors), create a custom palette.

If necessary, recapture the video from a cleaner source to allow for better compression, or use a different codec or custom palette. Don't confuse poor performance with poor visual quality. Increasing the frame rate, decreasing the keyframe interval, or choosing higher quality settings will improve the appearance of individual frames, but may worsen overall playback if they increase the data rate unacceptably. Poor appearance may be preferable to poor performance.

Audio track doesn't play

Test the video in an external video application. Ensure that it contains a soundtrack that is enabled (see the *trackCount()*, *trackType()*, *trackEnabled()*, and *setTrackEnabled()* functions).

Check that *the movieRate of sprite* or *mRate of sprite* is 1 and *the frameRate of member* is 0. Audio plays only if the video is set to *Synch to SoundTrack* mode. In D7, the sound in an audio-only QuickTime File won't be heard if the video is not played direct-to-Stage.

Ensure that other QT sounds play. Set *the soundLevel* to 7, *the sound of member* to `TRUE`, and *the volume of sprite* or *volumeLevel of sprite* to 255.

QTW2 and VFW may not recognize multiple soundtracks in a single digital video file. Mix the multiple tracks into a single track using SoundEdit 16 or use separate audio-only QT or AVI files.

Under Windows, you may need to stop all other sounds to free the sound device for QTW or VFW. Set *the soundKeepDevice* to `FALSE`, and see Chapter 15.

Sound is dropped out by Director in very low memory situations. Unload other cast members and do not preload the entire digital video.

Director runs out of memory

The default value for *the preLoadRAM*, 0, uses all available RAM when preloading DVs for which *the preLoad of member* is `TRUE`. Ensure that no cast members are using the "Never" or "Last" unload settings (*the purgePriority of member* should be 2 or 3, not 0 or 1).

Poor digital video performance

Test the video in MoviePlayer; if the movie is not compressed and interleaved properly, or plays poorly in MoviePlayer, it will only be worse in Director. Ensure that the DV file is not fragmented on disk and that there is adequate working disk space on the system disk.

QuickTime performs best in 16-bit (thousand of colors), although this can compromise performance of Director animations designed for other depths.

If the video plays back well in an external application, it should play adequately in Director. Test video playback in the Score by itself. Minimize other activities while playing videos. Set *the directToStage of member* to `TRUE`, and ensure that the DV sprite is not stretched, or is stretched only in 100% increments. Ensure that the *width of member*, *height of member*, *width of sprite*, *height of sprite*, *top of sprite,* and *left of sprite* properties are evenly divisible by 8. Do not use *on idle* handlers or other time-consuming Lingo while playing digital video. Reduce Director's frame rate in the Tempo channel to about 5 fps, so that Director spends less time trying to redraw the Stage.

The most common culprits of poor performance are an excessive data rate (see Table 9-3) and improper interleaving of the audio and video tracks. Some codecs are more processor-intensive.

If the video plays well from a hard drive, but not from a CD-ROM, specify a faster minimum CD-ROM speed drive or reduce the date rate of the digital video. Return to your DV compression program and increase the keyframe interval, reduce the frame rate to 10 to 15 fps, reduce the height and width (to, say, 320×240), or reduce the quality setting to lower the data rate. If necessary, recapture the video with a cleaner source to allow for better compression or use a different codec.

The QuickTime 3 Xtra uses DirectDraw under Windows in D6.5 when it is enabled. But DirectDraw causes problems with some video cards when using QTW3. The QuickTime Control Panel *Video Settings* defaults to *Options* (*Enable DCI*, *Enable DirectDraw*, *Enable DirectDraw Acceleration*). This causes the cursor to disappear and severely degrades performance of non-video elements. Change it to *Safe Mode (GDI Only)* to solve the problems (this doesn't adversely affect video playback outside Director).

If using QTW3, install DirectX. It improves performance dramatically. Director plays AVI files via the VFW interface, ActiveMovie, and DirectShow, but does not explicitly use the optimized DirectShow API.

Repeatedly setting *the volume of sprite* under Windows degrades performance, and repeatedly setting the *mRate* or *movieRate of sprite* may cause stuttering. Avoid setting the property if it has not changed, as shown in Example 16-7.

Example 16-7: Setting a Property Only When Needed

```
on exitFrame
  global gNewVolume, gVideoSprite
  -- Set the new value only if it represents a change
  if the volume of sprite gVideoSprite <> gNewVolume
    set the volume of gVideoSprite = gNewVolume
  end if
end
```

Video appears, but does not update or play

Ensure that Director is playing (don't use *pause*, use *go the frame*), and that *the movieTime of sprite* is 1. If looping in a *repeat* loop (not necessarily recommended), include *updateStage* within the repeat loop to redraw the video. Stop the video (set *the movieRate of sprite* to 0) before performing any transitions.

Video sprite conflicts or redraw problems

If a video is being played direct-to-Stage (which it usually should be), it will play in front of all other sprites and ignore ink effects, because it bypasses Director's offscreen compositing buffer. It will also leave trails. Refresh the screen as described under "Stage display options" earlier in this chapter.

To place other sprites over a DV sprite, set the DV to non-direct-to-Stage (not available with QTW2 for Windows). This is not recommended, because it severely degrades performance. Modify your interface instead.

Other Video and Non-Video Formats

The spectrum of QT3-supported formats and related digital video issues is enormous, and well beyond the scope of this book. In the following sections, I touch on the key points. More information can be found by searching the Macromedia TechNotes or visiting *http://www.zeusprod.com/nutshell/chapters/digvid.html*.

MPEG, Video CD, DVD, VideoDiscs

MPEG (Motion Picture Expert Group) is a standard for playing compressed audio and video data. MPEG 1 is designed for digital media that can supply data at rates up to 1.5 Mbits/sec, and MPEG 2 is a compatible extension of MPEG 1 designed for digital media that can supply data at rates of 4 to 10 Mbits/sec.

MPEG playback can be achieved via software alone or augmented with hardware decoders (necessary on less powerful computers). QuickTime supports software-only MPEG 1 playback on the Macintosh, provided the QuickTime MPEG Extension is installed. It is unclear whether QT4 will support software-only MPEG 2.

MPEG playback is not supported directly by QTW2 or QTW3. Under Windows MPEG video can be played using *mci* commands, the MPEG Xtra or DirectMedia Xtra both from Tabuleiro da Baiana Multimedia (*http://www.tbaiana.com*), or the MPEG ActiveX control (requires the ActiveX Xtra included with D6.5 and D7).

See Macromedia TechNote 03531, "MPEG Video FAQ," and the article on MPEG at *http://www.director-online.com*; see *http://www.zeusprod.com/nutshell/mpeg.html* for additional links.

Video CD (the so-called WhiteBook format) is a specification for playing MPEG 1 and MPEG 2 video and audio from a CD-ROM platter. Likewise, DVD-Video is an specification for playing MPEG 1 and MPEG 2 video and audio from a DVD-ROM platter. These are typically used in videodisc players.

Don't confuse DVD-Video with DVD-ROM. DVD-ROM is a storage medium (essentially a large CD-ROM) using the UFS (universal file system) supported by Windows 98 and DVD-ROM drives. The capacity varies with the media type (DVD-5, DVD-10, etc.). DVD-ROMs can be played back in any computer with the appropriate hardware and software (Real Magic's Hollywood board with a DVD-ROM drive is about $400).

DVD-Video is a video playback standard. Although DVD-Video discs allow some interactivity, they typically contain cinematic movies, such as *Titanic*, and are played on separate DVD-players attached to large-screen TVs. DVD-Video decoder cards are available for computers, but the DVD-Video format is not suitable for typical Director-style applications involving a lot of interactivity and a custom GUI. Controlling an external device such as a video disc player from Lingo usually requires an Xtra that is specifically designed to communicate with that device. Under Windows, you can issue *mci* commands if the device is MCI-compliant.

DVD-ROM burners ($15K) and media ($50 each) are currently very expensive, as are "one-off" pre-masters ($1,000), which you can obtain at a service bureaus in lieu of buying a burner. Creating the output files for a DVD-Video requires a separate authoring and compression system ($40–50K), such as the one from Sonic Solutions (*http://www.sonic.com*).

See *http://www.cinram.com* for details on these formats and media and see *http://www.zeusprod.com/nutshell/dvd.html* for additional links.

Director 7 does not add any new support for MPEG or DVD beyond the support in prior versions. The DirectMedia Xtra (*http://www.tbaiana.com*) will play. VOB files designed for DVD-Video discs.

QTVR and VRML

QTVR is just one of the many non-video formats supported by the larger QuickTime architecture. QTVR files should have "MooV" file type or .MOV extension, but unfortunately the same file type is used by standard QT movies.

There are two distinct types of QTVR files movies: so-called *panoramas* (also called *navigable* or *pano* movies), in which the user can explore a 360-degree view as if spinning in the middle of a room, and *object* movies in which the user can rotate an object and view it from all angles as if it were in his hands. The pano or object movies are created with software from Apple or third parties by warping and stitching one or more photographic images together. (See *http://www.apple.com/qtvr* for details.) Some rendering programs output QTVR files, too.

There are several ways to import QTVR content (but you should *not* import QTVR files as QT2-style cast members using `File ➤ Import` in D6.5 and earlier versions):

- In D6.5, use `Insert ➤ Media Element ➤ QuickTime 3` and then click *Browse* to choose a QTVR file. In D7, use `File ➤ Import`. Either of these imports the QTVR file as a QT3-style cast member and is recommended for QTVR 2.0 and later. See *the isVRMovie of member* property in Table 16-9 and the QTVR-specific properties in Table 16-10.
- Use a third-party Xtra such as the full-featured QTVR 2 Sprite Xtra (*http://www.glink.net.hk/~gemmay*). Obtain the latest version if using D7.
- Use the QTVR 1.0 Xtra included on both the Director 5 and Director 6 CDs. See the "QTVR 1.0 Xtra" later in this chapter.

All QTVR-related Xtras require a full QT installation, including the QTVR system omponents. Using QTVR files as *#quickTimeMedia* members requires the QT3 Asset Xtra and D6.5 or D7. Because the content visible in a QTVR movie changes as the user navigates, you can't use standard Director hotspots with QTVR. Hotspots are painted over the desired clickable areas in the stitched panoramic image just before it is turned into a QTVR movie. Tools including Apple's QTVR Authoring Studio will embed the hotspots in the QTVR movie when it is created (the hotspot colors become invisible). When the user clicks a hotspot, Director will receive the event, including the hotspot ID (equivalent to the hotspot's color index, 0 to 255) and pass it to the specified callback handlers, allowing you to take appropriate action.

A full discussion of QTVR is beyond the scope of this book. See the following resources and *http://www.zeusprod.com/chapters/qtvr.html* for more information.

QTVR, 3D, and VRML Xtras

Help on integrating QTVR into Director:

http://www.geocities.com/SiliconValley/Heights/6791/

QTVR 2 Sprite Xtra—full-featured:

http://www.glink.net.hk/~gemmay/

RealVR Xtras from RealSpace—QTVR and VRML:

http://www.rlspace.com

3D Dreams—Shells Interactive VRML Xtra for Shockwave:

http://www.shells-ifa.com/index.html

QTVR 1.0 Xtra

The obsolete QTVR Xtra is a Lingo *scripting* Xtra, not a Sprite Asset Xtra like the QT3 Asset Xtra. The QTVR Xtra does not "import" QTVR files or create QTVR cast members, but it accesses external QTVR 1.0 files via Lingo.

The QT3 Sprite Xtra can import QTVR 2.0 panorama or object movies, but it won't work with QTVR 1.0 movies nor support the oldest Macs and PCs. The older

QTVR Xtra supports QTVR 1.0 files on Director 5 and Director 6 (and supports 68K Macs and Windows 3.1, but does not work with Shockwave).

The QTVR Xtra gives finer control over QTVR 1.0 (i.e. it has more Lingo commands) than offered by the QT3 Sprite Xtra for QTVR 2.0 files. If using QTVR 2.0 files, consider the third-party QTVR 2 Sprite Xtra cited earlier in this chapter.

Table 16-18 shows only a few of the commands supported by the QTVR Xtra. The QTVR Xtra's full documentation and examples can be found under the *Macromedia:QTVR Xtra* folder on the D6 CD or printed in the Message windows using:

```
put mMessageList (xtra "QTVRXtra")
```

Notice that a QTVR 1.0 file is opened via Lingo, not stored as a cast member or used in the Score. Note also that the newer QT3 Asset Xtra has replaced many of the QTVR Xtra's function calls with properties. For example the QT3 Asset Xtra's *VRwarpMode of member* property replaces the QTVR Xtra's *QTVRGetWarpMode()* and *QTVRSetWarpMode()* functions. Furthermore, the QTVR Xtra often used strings, instead of the symbols or integers used by the QT3 Asset Xtra. For example, the QTVR Xtra's *QTVROpen* command requires the window dimensions as a string of four integers.

Table 16-18: QTVR 1.0 Xtra Commands

Command	**Notes**
QTVREnter (xtra "QTVRXtra")	Call once to initialize the Xtra.
set *instance* = new (xtra "QTVRXtra")	Create an instance.
put mMessageList (xtra "QTVRXtra")	Display documentation.
QTVRExit (xtra "QTVRXtra")	Close the Xtra.
QTVROpen(instance, "vrMovie.mov", "*left*,*top*,*right*,*bottom*", "visible" | "invisible")	Opens QTVR in specified rectangle; returns error string, or EMPTY for success.

QuickDraw 3D

QuickDraw 3D (QD3D) is another cool offshoot of Apple's QuickTime/QuickDraw technology that never caught fire. QD3D does amazing things such as manipulate 3D objects in real time and use a QT video as a texture for an object.

Because it is processor-intensive, it is supported only on PowerPCs and Windows 32-bit systems. It requires that substantial extra RAM be allocated to Director, and naturally requires the QuickDraw 3D System Extensions (QuickDraw 3D, QuickDraw 3D Viewer, QD3D Custom Elements, QuickDraw 3D IR, QuickDraw 3D RAVE, Apple QD3D HW Driver, and Apple QD3D HW Plug-in).

As with QTVR files, there are several options for importing QD3D content:

- Install the QuickDraw 3D Xtra included on the D5 and D6 CD and use `Insert ➤ Media Element ➤ Media Element ➤ QuickDraw 3D Model` to import a 3D model file (file type "3DMF"). This creates a custom cast member of type *#QD3D_Xtra*, whose supported properties are shown in Table 16-19.

- In D6.5, use `Insert ➤ Media Element ➤ QuickTime 3` and then click *Browse* to choose a QD3D (3DMF) file. In D7, use `File ➤ Import`. Either of these imports the QD3D file as a QT3-style cast member (requires the QuickTime Asset Xtra). Unfortunately, the QuickTime Asset Xtra doesn't support any meaningful commands or properties specific to QD3D assets.
- Use a third-party Xtra such as the Focus3's QD3D Xtra, which supports multiple simultaneous QD3D models on the Stage.

The QuickDraw 3D Xtra

The QuickDraw 3D Xtra is not installed automatically with Director. It has been orphaned, but is still offers good, clean, scandal-free fun. It prefers the older QD3D drivers (included in *Drivers* folder on the Director 6 CD).

The Apple QD3D drivers installer installs sample QD3D model files in the *Apple Extras* folder. Macromedia's sample DIR movie on the Director 6 CD includes the infamous spinning dinosaur. Macromedia's defunct Extreme 3D software can also create 3DMF models.

Note that a QD3D cast member's model file can be external (linked) or internal to the cast (embedded) as set under the *Modeling* tab in the *QD3D Properties* dialog box. For performance reasons, you should set *the directToStage of member* to `TRUE` (or use the *Direct To Stage* option under the *Rendering* tab in the *QD3D Properties* dialog box).

Table 16-19 lists the member and sprite properties supported by the QuickDraw 3D Xtra. None of these properties are supported when using QD3D models imported as *#quickTimeMedia* cast members using `Insert ➤ Media Element ➤ QuickTime 3` in D6.5 or `File ➤ Import` in D7.

Note that *the type of member* of QD3D cast members is *#QD3D_Xtra*, and that the external file's name is stored in the *modelFile of member* property, not *the fileName of member* as with most other cast member types. Although not shown, note that all Lingo symbols used by the QD3D Xtra begin with "#q3". See the HTML documentation included with the QuickDraw 3D Xtra for more details.

Table 16-19: QD3D Xtra Member and Sprite Properties

QD3D Member Properties	QD3D Sprite Properties
ambientBrightness, ambientColor, autoRotate, autoRotateAngle, backColor, backFacing, badge, buttonDistance, buttonMove, buttonRotate, buttonZoom, cameraAspectRatio, cameraDirection, cameraFOV, cameraHeight, cameraHither, cameraPosition. cameraInterest, cameraType, cameraUpVector, cameraWidth, cameraYon, controller, diffuseColor, directManipulation, directToStage, dragAndDrop, fillStyle, frame, interpolation, lightBrightness, lightColor, lightDirection, modelCenter, modelFile, modelSize, position, rotation, scale, shading, specularColor, specularCoeff, texture, textureType, type	autoRotate, autoRotateAngle, badge, buttonDistance, buttonMove, buttonRotate, buttonZoom, cameraAspectRatio, camera Direction, cameraFOV, cameraHeight, cameraHither, cameraPosition. cameraInterest, cameraType, cameraUpVector, cameraWidth, cameraYon, controller, directManipulation, dragAndDrop, frame, modelCenter, modelSize, position, rotation, scale

Index

Symbols and Numbers

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z