Table of Contents

Preface ............................................................................................................... xi

Part I: Director’s Core Components

Chapter 1—How Director Works ................................................................. 3
  Director’s Frame-Based Model ................................................................. 3
  Effects Channels ..................................................................................... 5
  Sprites ...................................................................................................... 6
  Scripts and the Script Channel ............................................................... 8
  Cast Members ......................................................................................... 8
  Drawing to the Stage .............................................................................. 9
  Lingo Versus the Score ......................................................................... 11
  Cast, Score, and Lingo Cooperation .................................................... 15
  How Director Runs Your Movie .......................................................... 18
  The Grand Scheme of Things .............................................................. 19

Chapter 2—Being More Productive ............................................................. 21
  Plan Ahead ............................................................................................ 21
  Hardware and Software You’ll Need ................................................... 24
  Configuring Your System ..................................................................... 29
  Mastering Director ................................................................................ 33
  Shortcuts in Director ............................................................................ 36
  OS Shortcuts and Tips .......................................................................... 48
### Chapter 3—The Score and Animation ........................................ 55
  - Animation Techniques ................................................................. 55
  - The Score ..................................................................................... 65
  - Effects Channels ......................................................................... 69
  - Being More Productive in the Score .......................................... 78
  - Score Lingo ................................................................................ 88
  - Preventing Problems ................................................................. 98

### Chapter 4—CastLibs, Cast Members, and Sprites ............. 101
  - Cast Libraries ............................................................................... 101
  - Importing, Inserting, and Creating Assets ................................. 111
  - Working with Cast Members ..................................................... 123
  - CastLib and Cast Member Lingo ................................................. 129

### Chapter 5—Coordinates, Alignment, and Registration Points ........................................ 148
  - Registration Points and Alignment .......................................... 148
  - Alignment in the User Interface .............................................. 163

### Chapter 6—The Stage and Movies-in-a-Window .......... 166
  - The Stage .................................................................................... 166
  - Movies-in-a-Window ................................................................. 169
  - Declaring and Using MIAWs .................................................... 174
  - Window Properties ................................................................. 176

### Part II: Delivery and Optimization

### Chapter 7—Cross-Platform and OS Dependencies .... 195
  - Planning Your Cross-Platform Strategy .................................. 195
  - Cross-Platform Differences ................................................... 199

### Chapter 8—Projectors and the Runtime Environment .... 217
  - Runtime Projectors ................................................................. 217
  - Analyzing the Runtime Environment ...................................... 230
  - Projectors (Runtime) Versus Director (Authoring) ................ 238
  - Projector Utilities ................................................................. 243
<table>
<thead>
<tr>
<th>Chapter 9—Memory and Performance</th>
<th>252</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disk Storage and Memory Management</td>
<td>252</td>
</tr>
<tr>
<td>Media Sizes</td>
<td>255</td>
</tr>
<tr>
<td>Data Throughput</td>
<td>262</td>
</tr>
<tr>
<td>Director Memory Budget</td>
<td>263</td>
</tr>
<tr>
<td>Cast Member Loading and Unloading</td>
<td>267</td>
</tr>
<tr>
<td>Memory Optimization</td>
<td>282</td>
</tr>
<tr>
<td>Performance</td>
<td>287</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 10—Using Xtras</th>
<th>296</th>
</tr>
</thead>
<tbody>
<tr>
<td>Types of Xtras</td>
<td>296</td>
</tr>
<tr>
<td>Loading and Registering Xtras</td>
<td>314</td>
</tr>
<tr>
<td>Including Xtras with a Projector</td>
<td>322</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 11—Shockwave and the Internet</th>
<th>328</th>
</tr>
</thead>
<tbody>
<tr>
<td>Getting Started with Shockwave</td>
<td>329</td>
</tr>
<tr>
<td>Shockwave Plug-ins and ActiveX Controls</td>
<td>330</td>
</tr>
<tr>
<td>Running a Shocked Movie on a Web Page</td>
<td>339</td>
</tr>
<tr>
<td>Uploading Shocked Files to a Web Server</td>
<td>342</td>
</tr>
<tr>
<td>Support, Preferences, and Xtras Folders</td>
<td>345</td>
</tr>
<tr>
<td>Streaming Playback</td>
<td>348</td>
</tr>
<tr>
<td>Shockwave Differences</td>
<td>349</td>
</tr>
<tr>
<td>Projectors That Access the Internet</td>
<td>353</td>
</tr>
<tr>
<td>Communicating with the Browser</td>
<td>356</td>
</tr>
<tr>
<td>Network Errors—netError()</td>
<td>359</td>
</tr>
<tr>
<td>New Shockwave Features in D7</td>
<td>361</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Part III: Multimedia Elements</th>
<th></th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Chapter 12—Text and Fields</th>
<th>369</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rich Text, Fields, and Bitmapped Text</td>
<td>369</td>
</tr>
<tr>
<td>Text Appearance and Attributes</td>
<td>370</td>
</tr>
<tr>
<td>Manipulating Text in the Interface</td>
<td>376</td>
</tr>
<tr>
<td>Text and Field Lingo</td>
<td>380</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 13—Graphics, Color, and Palettes</th>
<th>396</th>
</tr>
</thead>
<tbody>
<tr>
<td>Color-Related Lingo Commands</td>
<td>396</td>
</tr>
<tr>
<td>Graphics Types</td>
<td>397</td>
</tr>
<tr>
<td>Colors Schemes and Color Depths</td>
<td>408</td>
</tr>
</tbody>
</table>
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
About This Book

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](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.
Conventions Used in This Book

- **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](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:

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

means that the following are equivalent:

```plaintext
  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:

```plaintext
  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
Conventions Used in This Book

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):

```lingo
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):

```lingo
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.

```lingo
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.

```lingo
-- 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 event 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
Conventions Used in This Book

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 the 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/nutsbell/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/nutsbell/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 Shock-Machine (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 MIAW's (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.
New Features in Director 7

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:


- 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:

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

can be used instead of:

```plaintext
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:

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

can be used instead of:

```plaintext
x = getAt (exampleList, 1)
setAt (getAt(someList, 7), 4, "newValue")
```
New Features in Director 7

For many more examples and details, see Chapters 4 and 12, and http://www.zeusprod.com/nutsbell/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 multi-player 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.
Director Resources

- 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 Nutsbell and Lingo in a Nutsbell**

O’Reilly and Associates:


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


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

- [http://www.webreview.com/](http://www.webreview.com/)

**Macromedia**

Macromedia home page and mirror sites:

- [http://www.macromedia.com](http://www.macromedia.com)
- [http://www-euro.macromedia.com](http://www-euro.macromedia.com)
- [http://www-asia.macromedia.com](http://www-asia.macromedia.com)
Director Resources

Director 7 new features, upgrade policy, and online docs:
  http://www.macromedia.com/support/director/bow/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/bow/subjects/

Shockwave Developer Center:
  http://www.macromedia.com/shockwave/
  http://www.macromedia.com/support/director/bow/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/

  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
flp://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
flp://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

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
```
We'd Like to Hear from You

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

Shockwave (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

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:

booksquestions@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 Enibow, 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 Willgen, 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
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.

![Figure 4-1: Cast window](image)

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
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:

```lingo
member whichMember of castLib whichCast
```

or in D7 notation:

```lingo
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 the `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 \textit{SHARED.CST}. This can wreak havoc if \textit{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.

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|}
\hline
\textbf{Version} & \textbf{External CastLib Name} & \textbf{Notes} \\
\hline
Director 3.1.3 & Shared Cast (Mac), SHRD CST.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. \\
\hline
Director 4 & \textit{SHARED.DIR} (unprotected) or \textit{SHARED.DXR} (protected) & Cross-platform file format, but Shared Cast still resided in same folder with main movie. D7 will not update from D4. \\
\hline
Director 5 & \textit{<AnyName>.CST} (unprotected), \textit{<AnyName>.CXT} (protected), or \textit{<AnyName>.CCT} (Shockwave) & Multiple external castLibs allowed. CastLibs can use any name and can reside in any folder. \\
\hline
Director 6 and 7 & Same as Director 5. & Shockwave casts can be used locally or reside at any URL. \\
\hline
\end{tabular}
\end{table}

\textit{Table 4-1: Shared Cast Versus External CastLibs}

\textbf{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.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{image}
\caption{Figure 4-2: Adjusting references to linked castLibs}
\end{figure}
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

<table>
<thead>
<tr>
<th>Operation</th>
<th>Menu Command</th>
<th>Mac</th>
<th>Win</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open/Close Cast window</td>
<td>Window ➤ Cast</td>
<td>Cmd-3</td>
<td>Ctrl-3</td>
</tr>
<tr>
<td>Open more Cast windows</td>
<td>Window ➤ Cast ➤ CastLibName / Window ➤ New Window</td>
<td>Opt-click castLib pop-up</td>
<td>Alt-click castLib pop-up</td>
</tr>
<tr>
<td>Open unlinked external Cast</td>
<td>File ➤ Open or use Toolbar button.</td>
<td>Cmd-0</td>
<td>Ctrl-0</td>
</tr>
<tr>
<td>Open Behavior Library (D6)</td>
<td>Xtras ➤ Behavior Library</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>Open Java Behaviors (D6.5)</td>
<td>Xtras ➤ Behavior Library for Java</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>Open Library Palette (D7)</td>
<td>Window ➤ Library Palette or use Toolbar button</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>Create, link, remove, or modify castLibs in use</td>
<td>Modify ➤ Movie ➤ Casts</td>
<td>Cmd-Shift-C</td>
<td>Ctrl-Shift-C</td>
</tr>
</tbody>
</table>
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. Compressed movies and castLib files are different only in that compacting removes any deleted cast members and optimizes cast member order and Score notation. Compact files do 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

<table>
<thead>
<tr>
<th>Command</th>
<th>File Type</th>
<th>Compact</th>
<th>Compress</th>
<th>Protect</th>
<th>Replace Original</th>
<th>Batch</th>
</tr>
</thead>
<tbody>
<tr>
<td>File ➤ Save1,2</td>
<td>DIR or CST</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>File ➤ Save As3</td>
<td>DIR or CST</td>
<td>✓</td>
<td></td>
<td></td>
<td>Optional</td>
<td></td>
</tr>
<tr>
<td>File ➤ Save and Compact2</td>
<td>DIR or CST</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>File ➤ Save As Shockwave movie</td>
<td>DCR or CST</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>File ➤ Save as Java4</td>
<td>DJR</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>File ➤ Save All1,5</td>
<td>DIR or CST</td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Xtras ➤ Update Movies ➤ Update6</td>
<td>DIR or CST</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Xtras ➤ Update Movies ➤ Protect</td>
<td>DXR or CXT</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Xtras ➤ Update Movies ➤ Convert to Shockwave movie(s)</td>
<td>DCR or CCT</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>

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.


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

http://www.angelfire.com/il/sgkel/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/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

<table>
<thead>
<tr>
<th>Asset Type</th>
<th>Notes</th>
<th>Mac</th>
<th>Win</th>
</tr>
</thead>
<tbody>
<tr>
<td>All Files</td>
<td>Shows all asset types</td>
<td>Shows only recognized file types</td>
<td>Imports unknown types as OLE</td>
</tr>
<tr>
<td>Bitmap Image1,2</td>
<td>All supported graphical types (including PICTs; JPEG import requires QuickTime)</td>
<td>.BMP (<code>BMP</code>), .GIF (<code>GIF</code>), .JPEG (<code>JPEG</code>), .LGR, .PCT, .PIC, .PICT (<code>PICT</code>), .PNT (<code>PNTG</code>), .PSD (<code>8BPS</code>), .TGA (<code>TPIC</code>), .TIF (<code>TIFF</code>)</td>
<td>.BMP, .DIB, .EPS, .FCC, .FCI, .GIF, .JPEG, .LGR, .PCD, .PCT, .PCX, .PIC, .PICT, .PNG, .PNT, .PSD, .TIF, .TGA, .WMF</td>
</tr>
<tr>
<td>PICT</td>
<td>PICTs only</td>
<td>.PCT, .PIC, .PICT (<code>PICT</code>)</td>
<td>.PCT, .PIC, .PICT</td>
</tr>
<tr>
<td>Palette1,2</td>
<td>Create palette in deBabelizer or Photoshop</td>
<td>PAIL (<code>8BCT</code>), imported with bitmap2</td>
<td>.PAL, imported with bitmap2 Photoshop CLUT</td>
</tr>
<tr>
<td>Scrapbook</td>
<td>Mac only</td>
<td><code>scbk</code></td>
<td>N/A</td>
</tr>
<tr>
<td>PICS</td>
<td>Mac only</td>
<td>.PICS (<code>PICS</code>)</td>
<td>N/A</td>
</tr>
<tr>
<td>FLC and FLI</td>
<td>Win only (AutoCAD)</td>
<td>N/A</td>
<td>.FLC, .FLI</td>
</tr>
<tr>
<td>Sound3</td>
<td>Supports uncompressed and IMA-compressed sounds</td>
<td>.AIF, .AIFC (<code>AIFC</code>), .WAV, .WAVE (<code>WAVE</code>), System 7 SND (<code>sndl</code>), .au (<code>ULAW</code>), .SWA (<code>SWA7</code>), .MP3 (<code>MP3</code>), .TGA, .TMP</td>
<td>.AIF, .AIFC, .WAV, .WAVE, .au, .SWA, .MP3</td>
</tr>
<tr>
<td>Director movie4</td>
<td>Director 5, 6, or 7 movie files</td>
<td>.DIR (<code>MV07</code>, <code>MV97</code>, <code>MV95</code>), .DCR (<code>FGDM</code>), .DXR (<code>M!07</code>, <code>M!97</code>, <code>M!95</code>)</td>
<td>.DIR, .DCR, .DXR</td>
</tr>
<tr>
<td>Director Cast5</td>
<td>Director 5, 6, or 7 cast files</td>
<td>.CST (<code>MC07</code>, <code>MC97</code>, <code>MC95</code>)</td>
<td>.CST</td>
</tr>
<tr>
<td>Digital Video</td>
<td>Mac: QuickTime Win: Video Clip</td>
<td>.MOV (<code>MooV</code>), .MPG, .MPEG (<code>MPEG</code>), .AVI (<code>VFW</code>)</td>
<td>.MOV, .AVI, .MPG, .MPEG</td>
</tr>
<tr>
<td>Rich Text (D6)1,6</td>
<td>MS Word creates RTF files</td>
<td>.RTF (<code>RTF</code>), .TXT (<code>TEXT</code>), .HTML, .HTML</td>
<td>.RTF, .TXT, .HTML, .HTML</td>
</tr>
<tr>
<td>Animated GIF</td>
<td>New in D7</td>
<td>.GIF (<code>GIF</code>)</td>
<td>.GIF</td>
</tr>
<tr>
<td>Shockwave Flash</td>
<td>New in D7; use Insert menu in D6.5</td>
<td>.SWF (<code>SWFL</code>)</td>
<td>.SWF</td>
</tr>
</tbody>
</table>

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 WAV 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 them 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**

<table>
<thead>
<tr>
<th>Asset Type</th>
<th>To Add Element</th>
<th>Mac</th>
<th>Win</th>
</tr>
</thead>
<tbody>
<tr>
<td>ActiveX1,2</td>
<td>Insert ➤ Control ➤ ActiveX</td>
<td>N/A</td>
<td>.OCX</td>
</tr>
<tr>
<td>Shockwave Audio3,4,5</td>
<td>Insert ➤ Media Element ➤ Shockwave Audio</td>
<td>.SWA ('SwaT')</td>
<td>.SWA</td>
</tr>
<tr>
<td>Other Sound formats</td>
<td>Copy Sound Edit 16 and scrapbook sounds to clipboard and paste into the Cast window</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Custom Cursor1</td>
<td>Insert ➤ Media Element ➤ Cursor</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>OLE6</td>
<td>Insert ➤ Media Element ➤ OLE Object</td>
<td>N/A</td>
<td>Various</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Media Type</th>
<th>Notes</th>
<th>Linked?</th>
</tr>
</thead>
<tbody>
<tr>
<td>#ActiveX¹</td>
<td>Requires ActiveX control included with D6.5 and D7 (Windows only).</td>
<td>Yes</td>
</tr>
<tr>
<td>#alpha¹</td>
<td>Requires Alphamania Xtra (<a href="http://www.medialab.com">http://www.medialab.com</a>).</td>
<td>No</td>
</tr>
</tbody>
</table>

**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 (continued)

<table>
<thead>
<tr>
<th>Media Type</th>
<th>Notes</th>
<th>Linked?</th>
</tr>
</thead>
<tbody>
<tr>
<td>#animGIF</td>
<td>New in D7. See Insert ➤ Media Element ➤ Animated GIF, and File ➤ Import.</td>
<td>Optional</td>
</tr>
<tr>
<td>#bitmap</td>
<td>Only unlinked images can be edited in Paint window.</td>
<td>Optional</td>
</tr>
<tr>
<td>#btned</td>
<td>Requires Custom Button Editor Xtra. Source bitmaps used for custom button can be deleted. Obsolete in D7.</td>
<td>No</td>
</tr>
<tr>
<td>#button</td>
<td>See buttonType of member (#checkBox, #pushButton, #radioButton).</td>
<td>No</td>
</tr>
<tr>
<td>#cursor</td>
<td>Requires Custom Cursor Xtra included with D6.5 and D7.</td>
<td>No</td>
</tr>
<tr>
<td>#digitalVideo</td>
<td>QT2.x or AVI files in D6. In D7, used only for AVI files under Windows. See digitalVideoType of member (#quickTime or #videoForWindows).</td>
<td>Yes²</td>
</tr>
<tr>
<td>#empty</td>
<td>Unoccupied cast member.</td>
<td>N/A</td>
</tr>
<tr>
<td>#field</td>
<td>See boxType of member (#adjust, #fixed, #limit, #scroll).</td>
<td>No</td>
</tr>
<tr>
<td>#filmloop</td>
<td>Cast members used by film loops must be retained.</td>
<td>No</td>
</tr>
<tr>
<td>#flash</td>
<td>Requires Flash Asset Xtra included with D6.5 and D7.</td>
<td>Optional</td>
</tr>
<tr>
<td>#font</td>
<td>New in D7. See Insert ➤ Media Element ➤ Font.</td>
<td>No</td>
</tr>
<tr>
<td>#movie</td>
<td>Only linked Director movies become #movie cast members. If imported as unlinked, the components are imported as different types.</td>
<td>Yes</td>
</tr>
<tr>
<td>#ole</td>
<td>Windows-only; treated as #bitmap on Macintosh.</td>
<td>Yes</td>
</tr>
<tr>
<td>#palette</td>
<td>See palette of member and paletteRef of member in Table 13-8.</td>
<td>No</td>
</tr>
<tr>
<td>#picture</td>
<td>Use Import PICT file as PICT option. Can’t be edited in Paint window.</td>
<td>No</td>
</tr>
<tr>
<td>#PopupMenu</td>
<td>Requires PopUp Xtra (<a href="http://www.updatestage.com/xtras">http://www.updatestage.com/xtras</a>).</td>
<td>N/A</td>
</tr>
<tr>
<td>#QD3D_Xtra</td>
<td>Requires QD3D Xtra (included on D6 CD and free from Macromedia).</td>
<td>Optional</td>
</tr>
<tr>
<td>#QuickTimeMedia</td>
<td>Requires QuickTime 3 Asset Xtra included with D6.5 and D7.</td>
<td>Yes</td>
</tr>
<tr>
<td>#richtext</td>
<td>To find rich text members, search for members of type Text under Edit ➤ Find ➤ Cast Member in D6.5. Obsolete in D7. See #text.</td>
<td>No</td>
</tr>
<tr>
<td>#script</td>
<td>See scriptType of member (#score, #movie, #parent).</td>
<td>No</td>
</tr>
<tr>
<td>#shape</td>
<td>See shapeType of member (#line, #oval, #rect, #roundRect).</td>
<td>No</td>
</tr>
<tr>
<td>#sound</td>
<td>AIFF and WAVE sound files can be playing using sound playFile without a cast member reference. See also #SWA.</td>
<td>Optional</td>
</tr>
<tr>
<td>#SWA</td>
<td>Requires SWA Xtras.</td>
<td>Yes</td>
</tr>
<tr>
<td>#text</td>
<td>New in D7. Replaces #richtext and supersedes #field cast members. ( #text type also referred to #field cast members in D4.)</td>
<td>No</td>
</tr>
</tbody>
</table>
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.

---

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

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

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.

---

**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 your 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

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

\(^1\) 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—all 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:

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

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

```director
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, Quick-
Time 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, Custom 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. Simi-
larly, 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**

<table>
<thead>
<tr>
<th>Action</th>
<th>Command</th>
<th>Mac</th>
<th>Win</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cut, copy, paste, or edit cast members</td>
<td>Edit menu, or on context-sensitive menu</td>
<td>Ctrl-click</td>
<td>Right-click</td>
</tr>
<tr>
<td>Modify cast member properties1</td>
<td>Modify ➤ Cast Member ➤ Properties (see also Table 2-8)</td>
<td>Cmd-I</td>
<td>Ctrl-I</td>
</tr>
<tr>
<td>Modify cast member script</td>
<td>Modify ➤ Cast Member ➤ Script, or use Script button.</td>
<td>Cmd-‘ (apostrophe). Opt-Script button opens new script.</td>
<td>Ctrl-‘ (apostrophe). Alt-Script button opens new script.</td>
</tr>
<tr>
<td>Edit in appropriate internal media editor</td>
<td>Edit ➤ Edit Cast Member, or select thumbnail and press Return. Double-click thumbnail in Cast, Sprite Inspector, or Sprite Toolbar</td>
<td>None</td>
<td>Alt-E.M</td>
</tr>
</tbody>
</table>
### Table 4-8: Cast Window Shortcuts (continued)

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

130 Chapter 4 – CastLibs, Cast Members, and Sprites
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:

```lingo
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

<table>
<thead>
<tr>
<th>Lingo</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>the activeCastLib</td>
<td>Returns the number of the currently selected castLib. Buggy in D7.0, but fixed in D7.0.1.</td>
</tr>
<tr>
<td>castLib whichCast</td>
<td>Refers to a castLib within an expression, e.g.: put the name of castLib whichCast</td>
</tr>
<tr>
<td>the castLibNum of member whichMember</td>
<td>Returns the number of the castLib containing a particular cast member. (Read only.)</td>
</tr>
<tr>
<td>duplicate (member fromMember {of castLib fromCast} {, member toMember of castLib toCast})</td>
<td>Duplicates the specified cast member. Returns new cast member position.</td>
</tr>
<tr>
<td>erase (member whichMember {of castLib whichCast})</td>
<td>Deletes the specified cast member (dangerous!). Always returns 0.</td>
</tr>
<tr>
<td>the fileName of castLib whichCast</td>
<td>Returns the complete path to a castLib file.¹ Can be set for external castLibs.</td>
</tr>
<tr>
<td>findEmpty (member whichMember {of castLib whichCast})</td>
<td>Finds the next available cast member slot in a castLib. If you don’t specify a castLib, it assumes castLib 1, not the activeCastLib.</td>
</tr>
<tr>
<td>importFileInto member whichMember {of castLib whichCast}, fileNameOrURL</td>
<td>Imports an asset into a castLib. Not recommended at runtime, because it consumes memory.</td>
</tr>
<tr>
<td>member whichMember {of castLib whichCast}</td>
<td>Refers to a member within an expression, e.g., put the name of member 1 of castLib 1</td>
</tr>
<tr>
<td>member (whichMember, whichCast)</td>
<td>Refers to a member in D7 notation.</td>
</tr>
<tr>
<td>move (member fromMember {of castLib fromCast} {, member toMember of castLib toCast})</td>
<td>Moves the specified cast member, but does not update Score notation! Existing cast member in destination will be replaced. Returns new cast member position.</td>
</tr>
</tbody>
</table>
Chapter 4 – CastLibs, Cast Members, and Sprites

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

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

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.

Example 4-1: Listing Internal and External castLibs

```ln
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
```

132 Chapter 4 – CastLibs, Cast Members, and Sprites
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:

```lingo
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:

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

To check if a sprite is empty, use:

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

or:

```lingo
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:

```lingo
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:

```lingo
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:

```lingo
set the member of sprite (the currentSpriteNum) = ¬
    member whichMember
set the memberNum of sprite (the currentSpriteNum) = ¬
    someMemberNumber
```
In D7, you can use:

```verbatim
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.)

```verbatim
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:

```verbatim
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:

```verbatim
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 nonexistent 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)
pull 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:

```
pull the memberNum of sprite 5
-- 2
pull 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:

```lingo
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):

```lingo
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):

```lingo
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

<table>
<thead>
<tr>
<th>Media Type</th>
<th>Cast Member Properties</th>
<th>Sprite Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>All types (prior to D7)</td>
<td>castLibNum, fileName,(^{2}) height, loaded, media, mediaReady, member, memberNum, modified, name, number, picture, purgePriority, rect, regPoint, scriptText, size, type, width</td>
<td>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</td>
</tr>
<tr>
<td>New properties in D7</td>
<td>thumbnail</td>
<td>bgColor, blendLevel, color, endFrame, flipH, flipV, loc2, quad, rotation, scriptList, skew, startFrame, volume</td>
</tr>
</tbody>
</table>
### Table 4-10: Cast Member and Sprite Properties (continued)

<table>
<thead>
<tr>
<th>Media Type</th>
<th>Cast Member Properties</th>
<th>Sprite Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>#ActiveX²</td>
<td>Each imported ActiveX control has its own custom member properties.</td>
<td>Each imported ActiveX control has its own custom sprite properties.</td>
</tr>
<tr>
<td>#animGIF³</td>
<td>directToStage, linked, fixedRate, playbackMode</td>
<td>See common properties.</td>
</tr>
<tr>
<td>#bitmap⁴</td>
<td>alphaThreshold,³ depth, dither,³ palette, paletteRef, picture, useAlpha³</td>
<td>See common properties.</td>
</tr>
<tr>
<td>#bttned² (obsolete in D7)</td>
<td>behavesLikeToggle, enabled, initialToggleState, labelString</td>
<td>behavesLikeToggle, enabled, isToggle</td>
</tr>
<tr>
<td>#button</td>
<td>alignment, backColor, buttonType, font, fontSize, fontStyle, foreColor, hilite, lineHeight, text</td>
<td>See common properties.</td>
</tr>
<tr>
<td>#cursor²</td>
<td>automask, cast memberList, cursorSize, hotspot, interval, type</td>
<td>N/A (see the cursor of sprite for other sprite types)</td>
</tr>
<tr>
<td>#digitalVideo (used for QT2 in D6, and AVI only in D7)</td>
<td>center, controller, crop, cuePointNames, cuePointTimes, digitalVideoType, directToStage, duration,⁵ frameRate, loop, pausedAtStart, preload, sound,⁶ startTime, stopTime, track, tracks, timeScale, video</td>
<td>currentTime,⁷ mostRecentCuePoint,⁷ movieRate, movieTime, startTime, stopTime, volume⁷</td>
</tr>
<tr>
<td>#empty</td>
<td>memberNum, number, name, type (other common properties not supported)</td>
<td>See common properties (most evaluate to zero for empty sprites).</td>
</tr>
<tr>
<td>#field⁸</td>
<td>alignment, autoTab, backColor, border, boxDropShadow, boxType, dropShadow, editable, font, fontSize, fontStyle, foreColor, lineCount, lineHeight, margin, pageHeight, picture,⁹ rect, scrollTop, text, wordWrap</td>
<td>editable, rect</td>
</tr>
<tr>
<td>#filmloop</td>
<td>center, crop, loop, media, sound, regPoint (read-only)</td>
<td>See common properties.</td>
</tr>
<tr>
<td>#flash²,¹⁰</td>
<td>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, viewW</td>
<td>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, viewW</td>
</tr>
</tbody>
</table>
### Cast Members and Sprite Properties

<table>
<thead>
<tr>
<th>Media Type</th>
<th>Cast Member Properties</th>
<th>Sprite Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>#font³</td>
<td>bitmapSizes, characterSet, font, fontStyle, name, originalFont, height, width</td>
<td>N/A (never used as a sprite)</td>
</tr>
<tr>
<td>#movie</td>
<td>center, crop, loop, scriptsEnabled, sound</td>
<td>See common properties.</td>
</tr>
<tr>
<td>#ole</td>
<td>See common properties.</td>
<td>See common properties.</td>
</tr>
<tr>
<td>#palette</td>
<td>See common properties (palette and paletteRef not supported).</td>
<td>N/A (never used as a sprite)</td>
</tr>
<tr>
<td>#picture</td>
<td>picture (palette and paletteRef not supported).</td>
<td>See common properties.</td>
</tr>
<tr>
<td>#QD3D_xtra²</td>
<td>See Table 16-19.</td>
<td>See Table 16-19.</td>
</tr>
<tr>
<td>#QuickTime-Media2,11</td>
<td>center, controller, crop, cuePointNames, cuePointTimes, directToStage, duration, fileName, frameRate, invertMask, isVRmovie, loop, mask, pausedAtStart, preload, regPoint, rotation, scale, sound, timeScale, translation, type, video</td>
<td>currentTime,² duration, isVRmovie, loopBounds, mostrecent CuePoint,² mouseLevel, movieRate, movieTime, mRate, mTime, rotation, scale, start-Time, stopTime, timeScale, translation, volume,² volumeLevel, VRFiel 0 View, VRHotSpotEnterCallback, VRHotSpotExitCallback, VRmotionQuality, VRmovedCallback, VRnode, VRnodeEnterCallback, VRnodeExitCallback, VRnodeType, VRpan, VRstaticQuality, VRtilt, VRTriggerCallback, VRWarpMode</td>
</tr>
<tr>
<td>#richtext (obsolete in D7)</td>
<td>pageHeight, picture, scrollTop, text (authoring only) (other properties available for #field members are not supported)</td>
<td>See common properties.</td>
</tr>
<tr>
<td>#script</td>
<td>scriptText, scriptType</td>
<td>Scripts themselves are never sprites, but can be attached to sprites (see the scriptNum, scriptList, and scriptInstanceList of sprite properties for other sprite types).</td>
</tr>
<tr>
<td>#shape</td>
<td>filled, lineDirection,³ lineSize, pattern, shapeType</td>
<td>blend, blendLevel, lineSize</td>
</tr>
<tr>
<td>#sound</td>
<td>channelCount, cuePointNames, cuePointTimes, loop, sampleRate, sampleSize</td>
<td>currentTime of sound,² mostRecent CuePoint of sound,² volume of sound²</td>
</tr>
</tbody>
</table>

---

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

<table>
<thead>
<tr>
<th>Media Type</th>
<th>Cast Member Properties</th>
<th>Sprite Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>#SWA²</td>
<td>bitsPerSample, bitRate, copyrightInfo, cuePointNames, cuePointTimes, duration,⁵ numChannels, percentPlayed, percentStreamed, preloadBuffer, preloadTime, sampleRate, soundChannel, state, streamName, url, volume</td>
<td>currentTime,⁷ mostRecentCuePoint,⁷ volume⁷</td>
</tr>
<tr>
<td>#text³</td>
<td>alignment, alpha, antiAlias, antiAliasThreshold, autoTab, backgroundColor, 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, useHypertextStyles, wordWrap</td>
<td>editable, rect</td>
</tr>
<tr>
<td>#transition</td>
<td>changeArea, chunkSize, duration,⁵ transitionType</td>
<td>N/A (never used as a sprite)</td>
</tr>
<tr>
<td>#vectorShape³</td>
<td>antiAlias, backgroundColor, closed, endColor, fillColor, fillCycles, fillDirection, fillMode, fillOffset, fillScale, gradientType, strokeColor, strokeWidth, vertexList</td>
<td>See also #flash entry.</td>
</tr>
<tr>
<td>#xtra</td>
<td>Xtra-dependent. See common properties. Xtra-dependent. See common properties.</td>
<td>Xtra-dependent. See common properties.</td>
</tr>
</tbody>
</table>

1. See the streamName of member and url of member properties for SWA cast members.
2. Requires an Xtra.
4. The hilite of member property does not apply to bitmaps and does not coincide with the Hilite 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**

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

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

```lingo
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

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**

```lingo
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**

```lingo
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
            importFileInto (castLib i, member j, the fileName of member j, 1)
            set errCode = status (fileObj)
            if errCode = 0 then
                if getLength (fileObj) / 1024.0 < 1024 then
                    importFileInto (castLib i, member j, the fileName of member j, 1)
                else
                    return errCode
                end if
            else
                return errCode
            end if
        end repeat
    end repeat
end importLinks
```
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 then
                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
            end if
        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 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
```
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

<table>
<thead>
<tr>
<th>Processor1</th>
<th>D7 Minimum/Preferred</th>
<th>D6.0.2 Minimum/Preferred</th>
<th>D5.0.1 Minimum/Preferred</th>
</tr>
</thead>
<tbody>
<tr>
<td>68K Mac</td>
<td>N/A</td>
<td>2,048/6,144 KB</td>
<td>2,048/4,096 KB</td>
</tr>
<tr>
<td>PowerMac with VM on</td>
<td>4,096/6,144 KB</td>
<td>2,048/6,1444 KB</td>
<td>2,048/4,096 KB</td>
</tr>
<tr>
<td>PowerMac with VM off</td>
<td>4,263/6,311 KB2</td>
<td>3,388/7,484 KB</td>
<td>2,986/5,034 KB</td>
</tr>
</tbody>
</table>

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.

---

254 Chapter 9 – Memory and Performance
**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 cast-Libs. 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:

\[
\text{(the width of member)} \times \text{(the height of member)} \times \text{(the depth of member)}/8
\]

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/nutsbell/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:

\[
\frac{\text{size of the external digital video file} \times \text{float(\text{the duration of member})}}{\text{(the digitalVideoTimeScale)}}
\]

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:

\[
\text{(samples per second)} \times \text{(bits per sample/8)} \times \text{(length in seconds)} \times \text{(number of channels)}
\]

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:

\[
\frac{(\text{bitRate of member})}{(8.192 \times \text{duration of member})}
\]

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**

<table>
<thead>
<tr>
<th>Media Type</th>
<th>Member Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>#bitmap</td>
<td>the depth, height, width, and size of member</td>
</tr>
<tr>
<td>#button</td>
<td><code>length(text of member)</code> and the size of member</td>
</tr>
<tr>
<td>#digitalVideo</td>
<td><code>duration, frameRate, and preLoad of member, and the preLoadRAM</code>; see Example 4-6</td>
</tr>
<tr>
<td>#field</td>
<td><code>length(text of member)</code> and the size of member</td>
</tr>
<tr>
<td>#filmLoop</td>
<td>the media of member (must be unwrapped)</td>
</tr>
<tr>
<td>#movie</td>
<td>see <code>getSize()</code> utility in Example 4-6</td>
</tr>
</tbody>
</table>
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

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

Table 9-4: CD-ROM Capacities

<table>
<thead>
<tr>
<th>Asset Type</th>
<th>Storage Requirement</th>
<th>Fits on CD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digital video (Cinepak—quarter screen)</td>
<td>400 KB/sec</td>
<td>27 minutes</td>
</tr>
<tr>
<td>Digital video (Cinepak—full screen)</td>
<td>1600 KB/sec</td>
<td>6.9 minutes</td>
</tr>
<tr>
<td>Digital video (Sorenson)</td>
<td>80 K/sec</td>
<td>135 minutes</td>
</tr>
<tr>
<td>MPEG-1 full-motion video</td>
<td>150 K/sec</td>
<td>74 minutes</td>
</tr>
<tr>
<td>MPEG-2 full-motion video</td>
<td>575 KB/sec</td>
<td>20 minutes</td>
</tr>
<tr>
<td>Audio (16-bit, 44.1 kHz, Stereo)</td>
<td>176 K/sec</td>
<td>64.5 minutes</td>
</tr>
<tr>
<td>Audio (16-bit, 22.050 kHz, Mono)</td>
<td>44 K/sec</td>
<td>4.3 hours</td>
</tr>
<tr>
<td>SWA (MP3) CD-quality</td>
<td>20 K/sec</td>
<td>9 hours</td>
</tr>
<tr>
<td>Bitmaps (640 × 480 × 256-color Director internal format)</td>
<td>200 KB/image1</td>
<td>3,300 images</td>
</tr>
<tr>
<td>Bitmaps (640 × 480 × millions of colors; JPEG compressed)</td>
<td>75 KB/image1</td>
<td>8,875 images</td>
</tr>
</tbody>
</table>

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.
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: Program Memory Budget

<table>
<thead>
<tr>
<th>Item</th>
<th>Mac</th>
<th>Win</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating system(^1)</td>
<td>5 to 10 MB</td>
<td>2 MB for Windows 3.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8-24 MB for Windows 95/98/NT</td>
</tr>
<tr>
<td>Projector code</td>
<td>1 to 3 MB</td>
<td>1 to 3 MB</td>
</tr>
<tr>
<td>Projector misc. memory</td>
<td>1 to 3 MB</td>
<td>1 to 3 MB</td>
</tr>
<tr>
<td>Offscreen buffer(^2)</td>
<td>300 KB or higher</td>
<td>300 KB or higher</td>
</tr>
<tr>
<td>Digital video drivers(^3)</td>
<td>500 KB to 1.5 MB</td>
<td>500 KB to 1 MB</td>
</tr>
<tr>
<td>Xtras(^4)</td>
<td>100 KB/Xtra</td>
<td>100 KB/Xtra</td>
</tr>
</tbody>
</table>

\(^1\) 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.

Table 9-6: Media Memory Budget

<table>
<thead>
<tr>
<th>Item</th>
<th>Mac</th>
<th>Win</th>
</tr>
</thead>
<tbody>
<tr>
<td>Score notation and cast member headers</td>
<td>100 KB to 1 MB</td>
<td>100 KB to 1 MB</td>
</tr>
<tr>
<td>Script cast members</td>
<td>100 KB to 1 MB</td>
<td>100 KB to 1 MB</td>
</tr>
<tr>
<td>Bitmaps and other cast members</td>
<td>2 MB to 3 MB</td>
<td>2 MB to 3 MB</td>
</tr>
<tr>
<td>Internal sound cast members</td>
<td>500 KB to 1 MB</td>
<td>500 KB to 1 MB</td>
</tr>
<tr>
<td>Digital video(^1)</td>
<td>500 KB to 1 MB</td>
<td>500 KB to 1 MB</td>
</tr>
</tbody>
</table>

\(^1\) 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.
Table 9-6: Media Memory Budget (continued)

<table>
<thead>
<tr>
<th>Item</th>
<th>Mac</th>
<th>Win</th>
</tr>
</thead>
<tbody>
<tr>
<td>Streaming buffer for external</td>
<td>400 KB</td>
<td>2.5 \times \text{the size of one second of audio}</td>
</tr>
<tr>
<td>sounds(^2)</td>
<td></td>
<td>(27 KB to 440 KB)</td>
</tr>
<tr>
<td>MIAW(^3)</td>
<td>500 KB to 1 MB</td>
<td>500 KB to 1 MB</td>
</tr>
</tbody>
</table>

\(^1\) Per typical 400 KB/sec video played concurrently. If the preload 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:

\[
\frac{(\text{the width of the stage} \times \text{the height of the stage})}{1024} \times \frac{\text{the colorDepth}}{8.0}
\]

For example, a 640\(\times\)480\(\times\)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.
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/nutsell/d7/globals.html](http://www.zeusprod.com/nutsell/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:

```lingo
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.

Table 9-7: Lingo Data Structure Memory Requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>Minimum Size</th>
<th>Max Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer</td>
<td>8 bytes</td>
<td>8 bytes</td>
</tr>
<tr>
<td>VOID</td>
<td>8 bytes</td>
<td>8 bytes</td>
</tr>
<tr>
<td>Symbol(^1)</td>
<td>8 bytes</td>
<td>8 bytes</td>
</tr>
<tr>
<td>Float</td>
<td>16 bytes</td>
<td>16 bytes</td>
</tr>
<tr>
<td>String</td>
<td>8 bytes + 1 byte per character</td>
<td>About 2 MB</td>
</tr>
<tr>
<td>List</td>
<td>8 bytes + size of elements</td>
<td>About 2 MB</td>
</tr>
<tr>
<td>Xtra instance</td>
<td>8 bytes + 180 bytes</td>
<td>No specific limit</td>
</tr>
<tr>
<td>Child object (script instance)</td>
<td>8 bytes + 180 bytes</td>
<td>No specific limit</td>
</tr>
</tbody>
</table>

\(^1\) Symbols always persist for the life of Director or the Projector. See Lingo Symbol Table Archaeology at [http://www.zeusprod.com/nutsell/chapters/symtable.html](http://www.zeusprod.com/nutsell/chapters/symtable.html).
**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 Nutsbell*)
- 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.
**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 window (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.

---

**Table 9-8: Unload Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>unload</code></td>
<td>Unloads only those cast members used in the Score.</td>
</tr>
<tr>
<td><code>unload fromFrame {, toFrame}</code></td>
<td>Unloads cast members in a range of frames, or in a single frame, if toFrame is omitted.</td>
</tr>
<tr>
<td><code>unload member fromMember, toMember</code></td>
<td>Unloads a range of cast members, as would the <code>unLoadMember</code> command.</td>
</tr>
<tr>
<td><code>unLoadMember</code></td>
<td>Unloads all cast members in D6. In D7.0, but not D7.0.1, you must manually specify a range.</td>
</tr>
<tr>
<td><code>unLoadMember member fromMember of castLib fromCast, {member toMember of castLib toCast}</code></td>
<td>Unloads a range of cast members or a single cast member if toMember is omitted1 (fromCast and toCast can be different castLibs).</td>
</tr>
<tr>
<td><code>unLoadMovie whichMovie</code></td>
<td>Unloads the specified movie (which can be a URL reference). The result returns 0 if successful or -1 if movie was not loaded.</td>
</tr>
</tbody>
</table>

---

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.
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

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

\begin{verbatim}
if the freeBlock < 200 * 1024 then
  unLoadMember
end if
\end{verbatim}

The entry under \textit{ramNeeded} in Macromedia’s \textit{Lingo Dictionary} includes an example that checks the \textit{ramNeeded()} against \textit{the freeBytes} to determine whether enough memory is available, but this is misleading. The \textit{ramNeeded()} overstates the required memory if some cast members are already loaded (but understates the RAM needed if preloading digital video). Furthermore, \textit{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.

\textbf{Reader Exercise}: Write a \textit{frameLoaded()} utility that checks whether the cast members needed for a frame are loaded. Model its syntax after the \textit{ramNeeded()} and \textit{frameReady()} functions. (Hint: use the \textit{loaded of member} property to check each sprite’s cast member. Use \textit{go frame} to move the playback head so that you can check frames other than the current frame.) \textit{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 (freeBytes = 26616908, 49200 bytes consumed)
   Member is in movie (or external castLib) "filename"
   File Seek Info: No file access occurred
   File Seek Info: Sought between files! Sought 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
      end if
    end repeat
  end repeat
end

Cast Member Loading and Unloading  275
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

display some statistics about what is still loaded
put "Total size of" && memberCount && " loaded member(s):" && memSize / 1024 && " KB"
pull "Including" && scriptCount && " script(s) totaling:" && scriptMemSize / 1024 && " KB"
pull "Including" && modCount && " modified member(s) totaling:" && modMemSize / 1024 && " KB"
pull "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 preload 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 \texttt{preLoadEventAbort} to \texttt{TRUE} to allow the user to interrupt preloading, or set the \texttt{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 \texttt{preLoad of member} is set. SWA files are preloaded according to the \texttt{preLoadTime of member} property using the \texttt{preLoadBuffer} command.

The preloading commands shown in Table 9-10 return a status in the \texttt{result} that is of dubious value in determining whether they succeeded. Entries preceded by “the” are properties, not commands.

\textit{Table 9-10: Preloading Commands}

<table>
<thead>
<tr>
<th>Command</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{downloadNetThing URL, localFile}</td>
<td>Downloads data from \texttt{URL}, so that it is local when needed.</td>
</tr>
<tr>
<td>\texttt{preLoad toFrame}</td>
<td>In D6 and D7.0.1, preloads all cast members used in the Score from the current frame to frame \texttt{toFrame}. In D7.0, use \texttt{preload the frame, toFrame} instead.</td>
</tr>
<tr>
<td>\texttt{preLoad fromFrame, toFrame}</td>
<td>Preloads all cast members used in the Score from frame \texttt{fromFrame} to \texttt{toFrame}.</td>
</tr>
<tr>
<td>\texttt{preLoad member fromMember, toMember}</td>
<td>Preloads a range of cast members, as would the \texttt{preLoadMember} command.</td>
</tr>
<tr>
<td>\texttt{the preLoad of member videoMember}</td>
<td>Determines whether a \texttt{#digitalVideo} or \texttt{#QuickTimeMedia} cast member’s external video file can be preloaded. See the \texttt{preLoadRAM} entry.</td>
</tr>
<tr>
<td>\texttt{preLoadBuffer member swaMember}</td>
<td>Preloads a portion of an SWA as set by the \texttt{preLoadTime of member}.</td>
</tr>
<tr>
<td>\texttt{preLoadCast}</td>
<td>Obsolete. See \texttt{preLoadMember}.</td>
</tr>
<tr>
<td>\texttt{the preLoadEventAbort}</td>
<td>If \texttt{TRUE}, allows a mouse click or key press to abort preloading. Defaults to FALSE.</td>
</tr>
<tr>
<td>\texttt{preLoadMember}</td>
<td>Preloads all cast members in the current movie in D6. In D7, it preloads only the first castLib.</td>
</tr>
</tbody>
</table>
Chapter 9 – Memory and Performance

Cast Member Loading and Unloading

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.

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:

```lingo
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
Cast Member Loading and Unloading

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**

<table>
<thead>
<tr>
<th>Command</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>cancelIdleLoad loadTag</code></td>
<td>Aborts preloading of cast members with the specified <code>loadTag</code>.</td>
</tr>
<tr>
<td><code>finishIdleLoad loadTag</code></td>
<td>Finishes preload command previously tagged by <code>loadTag</code>.</td>
</tr>
<tr>
<td><code>on idle</code></td>
<td>The <code>on idle</code> handler is called during each idle event, but need not be used for idle loading.</td>
</tr>
<tr>
<td><code>the idleHandlerPeriod = numTicks</code></td>
<td>Increasing the <code>idleHandlerPeriod</code> reduces the frequency with which the <code>idle</code>, <code>mouseLeave</code>, <code>mouseEnter</code>, and <code>mouseWithin</code> 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.)</td>
</tr>
<tr>
<td><code>idleLoadDone(loadTag)</code></td>
<td>Returns TRUE if cast members with specified <code>loadTag</code> are all loaded.</td>
</tr>
<tr>
<td><code>the idleLoadMode</code></td>
<td>Optionally allows idle loading (default is Never). See the next section.</td>
</tr>
<tr>
<td><code>the idleLoadPeriod = numTicks</code></td>
<td>Increasing the <code>idleLoadPeriod</code> increases the time between idle loads, allowing more time for idle activities other than cast member loading.</td>
</tr>
</tbody>
</table>
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

```lingo
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

```lingo
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/textcruncher/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:

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

is not as fast as:

```lml
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++. 

296
**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 window, 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

<table>
<thead>
<tr>
<th>Xtra Type</th>
<th>Accessed Via</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lingo Xtras1</td>
<td>Lingo only. See showXlib, the xtraList, interface(), and mMessageList() and Chapter 13 in Lingo in a Nutshell.</td>
</tr>
<tr>
<td>Sprite Xtras1</td>
<td>Insert ➤ Media Element, Insert ➤ Control, or File ➤ Import.</td>
</tr>
<tr>
<td>Transition Xtras1</td>
<td>Transition dialog box along with built-in transition types. See Modify ➤ Frame ➤ Transition.</td>
</tr>
<tr>
<td>MIX Xtras1</td>
<td>File ➤ Import and importFileInto. See also File ➤ Preferences ➤ Editors.</td>
</tr>
<tr>
<td>Export and Import Xtras</td>
<td>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.</td>
</tr>
<tr>
<td>Tool Xtras</td>
<td>Xtras menu (authoring mode only).</td>
</tr>
<tr>
<td>Shockwave Audio1</td>
<td>Insert ➤ Media Element, File ➤ Import (in D7), and Xtras menus.</td>
</tr>
<tr>
<td>Photoshop Filters</td>
<td>Xtras ➤ Filter Bitmap and Xtras ➤ Auto Filter.</td>
</tr>
<tr>
<td>Sound Mixer Xtras1</td>
<td>the soundDevice, the soundDeviceList</td>
</tr>
</tbody>
</table>

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 Filler, 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, M107, FCDM, or MC07), regardless of their extensions or names. Director 5 and 6 for Macintosh used the older File Types MV97, M197, M*97, MV95, M195, 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 includes 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.primearithmetic.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/textcruncher/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):

**Utility Xtras**

The following Xtras have many OS-level functions for each platform.

Buddy API (cross-platform):
DirectOS Xtra (Windows only):
   http://www.directxtras.com/do_doc.htm
OSutil Xtra (Macintosh version is shipping, Windows version is imminent):

**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:#extraType` 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.

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 \texttt{#default} are included with D7 Projectors, unless manually removed from the list of Xtras under \textit{Modify} \texttt{\textgreater} \textit{Movie} \texttt{\textgreater} \textit{Xtras}.

\textit{Table 10-3: Graphic MIX Xtras by Platform}

<table>
<thead>
<tr>
<th>D6 Type$^1$</th>
<th>D7 Type</th>
<th>PowerPC</th>
<th>Mac 68K</th>
<th>Win 32</th>
<th>Win 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>#service</td>
<td>#default</td>
<td>Mix Services</td>
<td>Mix Services</td>
<td>MixServices.X32 (D7) mix32.X32 (D6)</td>
<td>Mix16.X16</td>
</tr>
<tr>
<td>#mix$^2$</td>
<td>N/A</td>
<td>BMP Import Export</td>
<td>BMP Import Export</td>
<td>BMP Import Export</td>
<td>MixBMP.X16</td>
</tr>
<tr>
<td>#mix$^2$</td>
<td>N/A</td>
<td>GIF Export</td>
<td>None</td>
<td>GIF Export.X32</td>
<td>None</td>
</tr>
<tr>
<td>#mix</td>
<td>#default</td>
<td>GIF Import</td>
<td>GIF Import 68k</td>
<td>GIF Import.X32</td>
<td>MixGIF.X16</td>
</tr>
<tr>
<td>#mix$^3$</td>
<td>N/A</td>
<td>None</td>
<td>None</td>
<td>ImageMark Import.X32</td>
<td>None</td>
</tr>
<tr>
<td>#mix$^2$</td>
<td>N/A</td>
<td>JPEG Export</td>
<td>None</td>
<td>JPEG Export.X32</td>
<td>None</td>
</tr>
<tr>
<td>#mix$^2$</td>
<td>#default</td>
<td>JPEG Import</td>
<td>JPEG Import 68k</td>
<td>JPEG Import.X32</td>
<td>MixJPEG.X16</td>
</tr>
<tr>
<td>#mix</td>
<td>N/A</td>
<td>LRG Import Export</td>
<td>LRG Import Export</td>
<td>LRG Import Export.X32</td>
<td>MixLRG.X16</td>
</tr>
<tr>
<td>#mix</td>
<td>N/A</td>
<td>MacPaint Import</td>
<td>MacPaint Import</td>
<td>MacPaint Import.X32</td>
<td>MixMcPnt.X16</td>
</tr>
<tr>
<td>#mix</td>
<td>N/A</td>
<td>Palette Import</td>
<td>Palette Import</td>
<td>Palette Import.X32</td>
<td>MixPal.X16</td>
</tr>
<tr>
<td>#mix</td>
<td>N/A</td>
<td>Photoshop 3.0 Import</td>
<td>Photoshop 3.0 Import</td>
<td>Photoshop 3.0 Import.X32</td>
<td>MixPS30.X16</td>
</tr>
<tr>
<td>N/A</td>
<td>N/A</td>
<td>Photoshop™ Filters</td>
<td>None</td>
<td>Pshopflt.X32</td>
<td>None</td>
</tr>
<tr>
<td>#mix$^4$</td>
<td>N/A</td>
<td>None</td>
<td>None</td>
<td>Photoshop Clut Import.X32$^4$</td>
<td>None</td>
</tr>
<tr>
<td>#mix$^2$</td>
<td>N/A</td>
<td>PICT Import Export</td>
<td>PICT Import Export</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>#mix</td>
<td>N/A</td>
<td>PNG Import Export</td>
<td>PNG Import Export</td>
<td>PNG Import Export.X32</td>
<td>MixPng.X16</td>
</tr>
<tr>
<td>#mix</td>
<td>N/A</td>
<td>Targa Import Export</td>
<td>Targa Import Export</td>
<td>Targa Import Export.X32</td>
<td>MixTARGA.X16</td>
</tr>
<tr>
<td>#mix</td>
<td>N/A</td>
<td>TIFF Import Export</td>
<td>TIFF Import Export</td>
<td>TIFF Import Export.X32</td>
<td>MixTIFF.X16</td>
</tr>
</tbody>
</table>

$^1$ The Xtra’s type is specified in \texttt{XTRAINFO.TXT} and can be modified as desired. The GIF Import and JPEG Import Xtras are listed twice (as both \texttt{#net} and \texttt{#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 Xtras 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](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.

---

**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:
- Swastng.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

<table>
<thead>
<tr>
<th>D6 Type</th>
<th>D7 Type</th>
<th>PowerPC</th>
<th>Mac 68K</th>
<th>Win 32</th>
<th>Win 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>#asset</td>
<td>Obsolete</td>
<td>Button Editor</td>
<td>Button Editor</td>
<td>Buttoned.X32</td>
<td>Buttoned.X16</td>
</tr>
<tr>
<td>#asset</td>
<td>Obsolete</td>
<td>QuickDraw 3D Xtra</td>
<td>None</td>
<td>QD3DXtra.X32</td>
<td>None</td>
</tr>
<tr>
<td>#asset</td>
<td>N/A</td>
<td>CursorsPPC (D6)</td>
<td>None</td>
<td>Cursor.X32</td>
<td>None</td>
</tr>
<tr>
<td>#asset</td>
<td>#default</td>
<td>Flash Asset PPC</td>
<td>None</td>
<td>Flash Asset.X32</td>
<td>None</td>
</tr>
<tr>
<td>#asset</td>
<td>#default</td>
<td>QuickTime Asset PPC (D6)</td>
<td>None</td>
<td>QuickTime Asset.X32 (D6); QT3Asset.X32 (D7)</td>
<td>None</td>
</tr>
<tr>
<td>#asset</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>ActiveX.X32</td>
<td>None</td>
</tr>
<tr>
<td>#lingo</td>
<td>N/A</td>
<td>FileIOXtraFat (D6)</td>
<td>FileIOXtraFat</td>
<td>FILEIO.X32</td>
<td>FILEIO16.X16</td>
</tr>
<tr>
<td>#lingo</td>
<td>Obsolete</td>
<td>QTVRXtra</td>
<td>QTVRXtra</td>
<td>QTVRW32.X32</td>
<td>QTVRW.X16</td>
</tr>
<tr>
<td>#lingo</td>
<td>Obsolete</td>
<td>N/A</td>
<td>N/A</td>
<td>DirMMX.X32</td>
<td>None</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Usage</th>
<th>Type</th>
<th>PowerPC</th>
<th>Win32</th>
</tr>
</thead>
<tbody>
<tr>
<td>Animated GIF sprite</td>
<td>N/A</td>
<td>Animated GIF Asset</td>
<td>Animated GIF Asset.X32</td>
</tr>
<tr>
<td>Animated GIF options¹</td>
<td>N/A</td>
<td>Animated GIF Options</td>
<td>Animated GIF Options.X32</td>
</tr>
<tr>
<td>Vector shapes¹,²</td>
<td>N/A</td>
<td>Shape Xtra</td>
<td>ShapeXtra.X32</td>
</tr>
<tr>
<td>Text creation¹</td>
<td>N/A</td>
<td>Text Asset Options</td>
<td>TextAuth.X32</td>
</tr>
<tr>
<td>Text sprites</td>
<td>#default</td>
<td>Text Asset PPC</td>
<td>TextAsset.X32</td>
</tr>
<tr>
<td>Text rendering</td>
<td>#default</td>
<td>TextXtra PPC</td>
<td>TextXtra.X32</td>
</tr>
<tr>
<td>XML parsing</td>
<td>N/A</td>
<td>XMLParser PPC Xtra</td>
<td>XMLParser.X32</td>
</tr>
<tr>
<td>DXR compression¹</td>
<td>N/A</td>
<td>LZ77 Compression PPC Xtra</td>
<td>LZCompr.X32</td>
</tr>
<tr>
<td>DCR compression¹</td>
<td>N/A</td>
<td>Squish Rules PPC Xtra</td>
<td>Squish.X32</td>
</tr>
<tr>
<td>Multiuser server</td>
<td>N/A</td>
<td>Multiusr</td>
<td>Multiusr.X32</td>
</tr>
</tbody>
</table>

³12 Chapter 10 – Using Xtras
### 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 Actor Control and Cast Control Xtras** are used 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.


- **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 in 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.updatstage.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 Macro-media'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`, `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 `Director5.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 (*jwin@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
Loading and Registering Xtras

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

<table>
<thead>
<tr>
<th>Platform</th>
<th>Xtras Folder</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows 3.1</td>
<td>C:\Windows\Macromedia\Xtras</td>
</tr>
<tr>
<td>Windows 95/98/NT</td>
<td>C:\Program Files\Common Files\Macromedia\Xtras</td>
</tr>
<tr>
<td>Macintosh</td>
<td>MacHD:System Folder:Macromedia:Xtras</td>
</tr>
</tbody>
</table>

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, Shock-Machine, 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.zeus-prod.com/nutsbell/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:Macro-media: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/nutsbell/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:

```lingo
put mMessageList (xtra "NetLingo")
```
or:

```lingo
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](image-url)
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 XTRAININFO.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 XTRAININFO.TXT file.

I prefer to delete the #default flag from the XTRAININFO.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 XTRAININFO.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 alertbook.) 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 XTRAININFO.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. Downsamped 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:

\[
\text{size in K} = \left(\frac{\text{bits per sample}}{8}\right) \times (\text{sample rate in kHz}) \times \left(\frac{\text{number of channels}}{\text{duration in seconds}}\right)
\]
For example, 22.050 kHz, 16-bit mono sound occupies:

\[ \frac{16}{8} \times 22.050 \times 1 = 44.1 \text{ K/sec} \]

And 44.100 kHz, 16-bit stereo sound occupies:

\[ \frac{16}{8} \times 44.100 \times 2 = 176.4 \text{ K/sec} \]

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):

\[ \frac{\text{bit rate in Kbps}}{8} = \text{data rate in K/sec} \]

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 K/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 ➤
Sound Playback in Director

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

<table>
<thead>
<tr>
<th>Feature</th>
<th>Macintosh</th>
<th>Windows</th>
</tr>
</thead>
<tbody>
<tr>
<td>Formats supported for import into Cast(^1)</td>
<td>SWA, AIFF, AIFC,(^2) WAVE, System 7 sounds, Sun AU,(^3) MPEG3</td>
<td>Same as Mac, except for System 7 sounds</td>
</tr>
<tr>
<td>Formats supported for external linking</td>
<td>Same as above (except System 7 sounds), plus QuickTime (AVI imported as QT3 only)</td>
<td>Same as above, plus QuickTime, AVI</td>
</tr>
<tr>
<td>Supported sampling rates(^4)</td>
<td>5.564, 7.418, 11.025, 11.127, 22.050, 22.255, 32.000, 44.100, and 48.000 MHz</td>
<td>11.025, 22.050, and 44.100 MHz</td>
</tr>
<tr>
<td>Multichannel audio</td>
<td>Built-in (zero latency)</td>
<td>Supported via MacroMix, DirectSound, or QT3Mix(^5) (non-zero latency)</td>
</tr>
<tr>
<td>Maximum number of sound channels</td>
<td>8</td>
<td>Up to 8 (set by MixMaxChannels in DIRECTOR.INI)</td>
</tr>
<tr>
<td>Multiple simultaneous audio sources</td>
<td>Yes</td>
<td>Not necessarily(^6)</td>
</tr>
<tr>
<td>System audio buffer size(^6)</td>
<td>System's audio buffer size is fixed</td>
<td>Settable via DIRECTOR.INI file.</td>
</tr>
<tr>
<td>the soundLevel (0 to 7)</td>
<td>Matches settings in Sound or Monitors &amp; Sound Control Panel</td>
<td>Matches SoundLevel settings in DIRECTOR.INI. See also Volume Control in Task Bar(^7)</td>
</tr>
</tbody>
</table>

\(^1\) 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, ULAU, 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>
<thead>
<tr>
<th>Method</th>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Score Sound channel</td>
<td>Easy and intuitive. No scripting. Can use Tempo channel to wait for sounds.</td>
<td>Limited control and poor synchronization; only two sound channels.</td>
</tr>
<tr>
<td>puppetSound</td>
<td>Control over sound triggering.</td>
<td>Sounds must be explicitly unpuppeted.</td>
</tr>
<tr>
<td>sound playFile</td>
<td>Sounds don’t require a cast member. Can be played in any sound channel.</td>
<td>Slows loading of video or graphic media.</td>
</tr>
<tr>
<td>Sound tracks in digital video cast member</td>
<td>Can start and stop sound at any point, or play sound fast, slow, or backward.</td>
<td>QT and AVI audio may not mix with other audio under Windows</td>
</tr>
<tr>
<td>SWA</td>
<td>High quality at low bandwidth (good compression). Accesses up to 8 sound channels.</td>
<td>Processor-intensive. Requires 68K with FPU, or Pentium for playback.</td>
</tr>
<tr>
<td>Flash Audio</td>
<td>Compact and integrated with Flash file.</td>
<td>Low quality. Conflicts with Director sounds.</td>
</tr>
</tbody>
</table>

Table 15-2: Audio Playback Method Comparison
Table 15-2: Audio Playback Method Comparison (continued)

<table>
<thead>
<tr>
<th>Method</th>
<th>Streamed</th>
<th>Loopable$^1$</th>
<th>Channels</th>
<th>Cue Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Score sound channels</td>
<td>Only if external</td>
<td>Only if internal</td>
<td>2</td>
<td>AIF, and WAVE (D6.5 or later)</td>
</tr>
<tr>
<td>puppetSound</td>
<td>Only if external</td>
<td>Only if internal</td>
<td>Up to 8</td>
<td>Same as above</td>
</tr>
<tr>
<td>sound playFile</td>
<td>Always</td>
<td>No</td>
<td>Up to 8</td>
<td>Same as above</td>
</tr>
<tr>
<td>Audio track in digital video</td>
<td>Always</td>
<td>No</td>
<td>Uses separate video mixer</td>
<td>Same as above</td>
</tr>
<tr>
<td>SWA</td>
<td>Only if external</td>
<td>Only if internal</td>
<td>Up to 8</td>
<td>Yes, for QT and AVI</td>
</tr>
<tr>
<td>Flash audio</td>
<td>N/A</td>
<td>No</td>
<td>N/A</td>
<td>No</td>
</tr>
<tr>
<td>MIDI</td>
<td>Varies</td>
<td>Yes</td>
<td>N/A</td>
<td>No</td>
</tr>
<tr>
<td>Third-party Xtras</td>
<td>Usually</td>
<td>Varies</td>
<td>Varies</td>
<td>Xtra-dependent$^2$</td>
</tr>
</tbody>
</table>

$^1$ 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.

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

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 #quickTime-Media) 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 \textit{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 \textit{movieRate} of \textit{sprite}. (Slower playback lowers the pitch of the audio track; faster playback raises the pitch.)

\textbf{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 \textit{Wait for Cue Point:End} option causes them to "play out."

\textbf{puppetSound}

Use \textit{puppetSound} to trigger a sound cast member (whether linked or unlinked) at an arbitrary time, such as in response to button-clicks or timeouts. \textit{PuppetSounds} played in channels 1 and 2 override the corresponding Score Sound channel. Use:

\begin{verbatim}
  puppetSound channel, "soundMemberName"
  puppetSound channel, member "whichMember" (of castLib whichCast)
\end{verbatim}

where \textit{channel} is a number from 1 to 8. If \textit{channel} is specified, the sound will trigger immediately. Otherwise, the default channel is 1, and the \textit{puppetSound} is not triggered until the playback head advances (or loops) or an \textit{updateStage} command is issued. Here, the default channel 1 is assumed:

\begin{verbatim}
  puppetSound "soundMemberName"
  puppetSound member "whichMember" (of castLib whichCast)
\end{verbatim}

Specifying 0 as the "sound" unpuppets a \textit{puppetSound}. Unpuppeting halts the current \textit{puppetSound}; unpuppetting channel 1 or 2 also returns control to the corresponding Score Sound channel. These commands unpuppet a sound channel:

\begin{verbatim}
  puppetSound channel, 0
  puppetSound 0 -- Unpuppets sounds channel 1 by default
\end{verbatim}

\textbf{Sound playFile}

The \textit{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. \textit{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 \textit{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

<table>
<thead>
<tr>
<th>Operation</th>
<th>Sound</th>
<th>SWA or MP3</th>
<th>QuickTime 3 or Digital Video</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type of member</td>
<td>#sound</td>
<td>#SWA</td>
<td>#digitalVideo or #quickTimeMedia</td>
</tr>
<tr>
<td>Number of channels in asset</td>
<td>channelCount of member</td>
<td>numChannels of member</td>
<td>trackCount(), trackType = #sound</td>
</tr>
<tr>
<td>Specify sound channel</td>
<td>Place in Score or puppetSound or sound playFile</td>
<td>soundChannel of member (0 assigns highest)</td>
<td>Allocated by OS</td>
</tr>
<tr>
<td>Volume¹</td>
<td>volume of sound</td>
<td>volume of member, volume of sprite</td>
<td>volume of sprite, sound of member, volumeLevel of sprite</td>
</tr>
<tr>
<td>Play in Score</td>
<td>Sound channel</td>
<td>sprite channel</td>
<td>sprite channel</td>
</tr>
<tr>
<td>Play via Lingo</td>
<td>sound playFile, puppetSound</td>
<td>play ()</td>
<td>movieRate of sprite, mRate of sprite³</td>
</tr>
<tr>
<td>Stop a sound</td>
<td>sound stop, puppetSound</td>
<td>stop(), pause()</td>
<td>set movieRate of sprite = 0, or set mRate of sprite = 0</td>
</tr>
<tr>
<td>Sound playing</td>
<td>soundBusy()</td>
<td>state of member = 3</td>
<td>movieRate of sprite, mRate of sprite, pausedAtStart of member</td>
</tr>
<tr>
<td>Wait for sound²</td>
<td>soundBusy()</td>
<td>state of member</td>
<td>movieRate, movieTime, mRate, or mTime of sprite³</td>
</tr>
<tr>
<td>Samples per second</td>
<td>sampleRate of member</td>
<td>sampleRate of member, bitRate of member</td>
<td>Not available</td>
</tr>
<tr>
<td>Bits per sample</td>
<td>sampleSize of member</td>
<td>bitsPerSample of member</td>
<td>Not available</td>
</tr>
<tr>
<td>Current position</td>
<td>currentTime of sound</td>
<td>percentPlayed, percentStreamed, and currentTime of sprite</td>
<td>movieTime, mTime, or currentTime of sprite</td>
</tr>
<tr>
<td>Length in seconds</td>
<td>See Example 15-3</td>
<td>duration of member × 1000</td>
<td>duration of member × the digitalVideoTimeScale</td>
</tr>
</tbody>
</table>

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).
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 a member of an SWA member is 0 (the default) it automatically sidesteps conflicts by using the highest available channel number.

Most Macintoshs 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.

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
```
version. For an overview of sound mixing under Windows, see Macromedia Tech-
Note #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 exclu-
sive 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](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 Tech-Note #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

A 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 Quick-
Time 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](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**

<table>
<thead>
<tr>
<th>Environment</th>
<th>MacroMix and DirectSound</th>
<th>QT3Mix</th>
</tr>
</thead>
<tbody>
<tr>
<td>D4, D5, SW4, SW5</td>
<td>WaveOut only1,2,3</td>
<td>N/A</td>
</tr>
<tr>
<td>D6.0.x, SW6.0</td>
<td>DirectSound1 (requires RSX) or WaveOut2,3</td>
<td>N/A</td>
</tr>
</tbody>
</table>
Sound Channels and Sound Mixing

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.


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\QT\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:

```plaintext
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 cannot 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`, `soundPlayFiles`, 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:

```plaintext
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 Channels and Sound Mixing

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.
Table 15-6: Sound-Related Interface Options

<table>
<thead>
<tr>
<th>Action</th>
<th>Command</th>
</tr>
</thead>
</table>
| 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 | 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.
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.

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-syncing 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 *isPast Cue Point()* 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](image)

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**

```lingo
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:

```lingo
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.

```lingo
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**

```lingo
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 \textit{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 \textit{Wait for Cue Point} option waits for a cue point in a Sound channel, digital video sprite, or SWA sprite. If the \textit{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 \textit{loop} option is set, but the \{Next\} setting will. The \textit{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.

\begin{verbatim}
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
\end{verbatim}

\textbf{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 \url{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:

```lingo
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:

```lingo
set numCues = count (the cuePointNames of member "mySound")
set lastCue = getLast (the cuePointTimes of member "mySound")
```

The *isPastCuePoint()* function takes the general form:

```lingo
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:

```lingo
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.
Cue Points and Timing

Table 15-7: Cue Point Functions

<table>
<thead>
<tr>
<th>Action</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wait for cue point</td>
<td>Tempo channel, isPastCuePoint(), or see Examples 15-6 and 15-7</td>
</tr>
<tr>
<td>Determine names of cue points</td>
<td>the cuePointNames of member</td>
</tr>
<tr>
<td>Determine times of cue points</td>
<td>the cuePointTimes of member</td>
</tr>
<tr>
<td>Determine whether a cue point has been reached</td>
<td>isPastCuePoint()</td>
</tr>
<tr>
<td>Check the last cue point passed for a sprite or sound channel</td>
<td>the mostRecentCuePoint of sprite, the mostRecentCuePoint of sound</td>
</tr>
<tr>
<td>Trigger an event when a cue point is reached</td>
<td>on cuePassed event handler (see Examples 15-6 and 15-7)</td>
</tr>
<tr>
<td>Identify the sprite triggering a cue point</td>
<td>See the the spriteNum of me or channelId passed to the on cuePassed handler</td>
</tr>
<tr>
<td>Forces preloading of list of cue points (highly undocumented and unsupported in D6.5, and removed in D7)</td>
<td>forcePreloadCuePoints (member whichMember)</td>
</tr>
</tbody>
</table>

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:

```plaintext
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 onCue-Passed 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

```plaintext
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, $8 \times \frac{1024}{1000}$) 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

<table>
<thead>
<tr>
<th>Delivery</th>
<th>Bit rate</th>
<th>Quality¹</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1 or CD-ROM</td>
<td>64–160 Kbps (8–20 K/sec)</td>
<td>Equal to source material</td>
</tr>
<tr>
<td>ISDN</td>
<td>48–56 Kbps (6–7 K/sec)</td>
<td>FM stereo to CD-quality audio</td>
</tr>
<tr>
<td>28.8–56 Kbps modem²</td>
<td>16–32 Kbps (2–4 K/sec)</td>
<td>FM mono or good quality AM</td>
</tr>
<tr>
<td>14.4 Kbps modem²</td>
<td>8 Kbps (1 K/sec)</td>
<td>Telephone</td>
</tr>
</tbody>
</table>

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:

$$(\text{the duration of member}) \times (\text{the bitRate of member}) / 8192$$

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*

<table>
<thead>
<tr>
<th>Property</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>the bitRate of member1</td>
<td>0 (not ready)</td>
</tr>
<tr>
<td>the bitsPerSample of member1,2</td>
<td>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.</td>
</tr>
<tr>
<td>the copyrightInfo of member1,2</td>
<td>Copyright text for sound file.</td>
</tr>
<tr>
<td>the cuePointNames of member1,2</td>
<td>List of names of cue points.</td>
</tr>
<tr>
<td>the cuePointTimes of member1,2</td>
<td>List of cue point times in milliseconds.</td>
</tr>
<tr>
<td>the currentTime of sprite1,2</td>
<td>Current point in playback in milliseconds.</td>
</tr>
<tr>
<td>the duration of member1</td>
<td>Duration of SWA file in seconds (different units than property of the same name for #digitalVideo, #quickTimeMedia, and #transition members).</td>
</tr>
<tr>
<td>getError (member swaMember)</td>
<td>Error status for SWA cast members. The integer value returned by getError() corresponds to the string returned by getErrorString(): 0: returns EMPTY string = &quot;&quot; 1: &quot;memory&quot; 2: &quot;network&quot; (or &quot;Network software error&quot;) 3: &quot;playback device&quot; 99: &quot;other&quot;</td>
</tr>
<tr>
<td>getErrorString (member swaMember)</td>
<td>See getError().</td>
</tr>
<tr>
<td>isPastCuePoint(member, cueID)</td>
<td>Returns a positive integer if the cue point has been passed.</td>
</tr>
<tr>
<td>the mediaReady of member</td>
<td>Indicates whether media has been completely downloaded. Appropriate only for internal nonstreaming SWA-compressed sounds.</td>
</tr>
<tr>
<td>the mostRecentCuePoint of member2</td>
<td>Returns number of most recent cue point passed.</td>
</tr>
<tr>
<td>the numChannels of member1,2</td>
<td>Number of channels (usually 1 or 2). Returns 1 when the bitRate of member is &lt;= 32000.</td>
</tr>
</tbody>
</table>

---

520 Chapter 15 – Sound and Cue Points
### 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

<table>
<thead>
<tr>
<th>Property</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>pause (member)</code></td>
<td>Pauses SWA stream, but not instantly.</td>
</tr>
<tr>
<td>the <code>percentPlayed</code> of member</td>
<td>Percentage of bytes played (0 to 100). Should be less than or equal to <code>percentStreamed</code>.</td>
</tr>
<tr>
<td>the <code>percentStreamed</code> of member</td>
<td>Percentage of bytes streamed from server (0 to 100).</td>
</tr>
<tr>
<td><code>play (member)</code></td>
<td>Begins playing SWA (initiates preload, too).</td>
</tr>
<tr>
<td><code>preLoadBuffer (member swaMember)</code></td>
<td>Begins preloading amount of data specified by the <code>preLoadTime</code> of member. Use <code>stop()</code> to “rewind” the media after preloading before using <code>play()</code>.</td>
</tr>
<tr>
<td>the <code>preLoadTime</code> of member</td>
<td>Duration of SWA audio (not download time) in seconds to be downloaded before playback begins (prevents skipping).</td>
</tr>
<tr>
<td>the <code>sampleRate</code> of member</td>
<td>Sample rate of original sound source (in Hz) before compression, such as 11025, 22050, or 44100.</td>
</tr>
<tr>
<td>the <code>soundChannel</code> of member</td>
<td>System sound channel in which to play SWA (0 uses highest available channel). Avoid using 1 and 2, the Score Sound channels.</td>
</tr>
<tr>
<td>the <code>state</code> of member</td>
<td>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</td>
</tr>
<tr>
<td><code>stop (member)</code></td>
<td>Stops SWA stream, but not instantly. Also rewinds SWA.</td>
</tr>
<tr>
<td>the <code>streamName</code> of member</td>
<td>URL of SWA file. Can be local. (Same as <code>url</code> of member.)</td>
</tr>
<tr>
<td>the <code>url</code> of member</td>
<td>URL of SWA file. Can be local. (Same as <code>streamName</code> of member.)</td>
</tr>
<tr>
<td>the <code>volume</code> of member</td>
<td>Volume of SWA, from 0 to 255.</td>
</tr>
</tbody>
</table>

1. Returns 0, [], EMPTY, or meaningless data unless the `state` of member is 2, 3, 4, or 5.
2. Appears for the first 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.

---

**Table 15-9: Shockwave Audio Lingo (continued)**
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](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*

<table>
<thead>
<tr>
<th>Command</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>beep (n)</td>
<td>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.</td>
</tr>
<tr>
<td>beepOn</td>
<td>If TRUE, Director beeps when the user clicks on an inactive sprite (one without a mouse script attached). Useful for debugging.</td>
</tr>
<tr>
<td>sound close channel</td>
<td>Closes the specified sound channel. This requires the sound buffer to be reallocated for the next sound.</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Command</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>sound stop</td>
<td>Stops the sound playing in the specified channel.</td>
</tr>
<tr>
<td>the soundEnabled</td>
<td>If FALSE, all sounds are muted.</td>
</tr>
<tr>
<td>the soundLevel</td>
<td>System volume from 0 (mute) to 7 (loudest).</td>
</tr>
<tr>
<td>the multiSound</td>
<td>If TRUE, the computer supports stereo sound.</td>
</tr>
</tbody>
</table>

Table 15-10: System-Level Sound-Related Lingo (continued)
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.

---

### Table 15-11: DIRECTOR.INI File Sound-Related Settings (continued)

<table>
<thead>
<tr>
<th>Command</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLLname (D6.5 only)</td>
<td>Default is MacroMix.DLL. Change to QT3Mix.DLL to use QT3 Sound Mixing (requires D6.5 with Service Pack).</td>
</tr>
<tr>
<td>rsxDontUseDirectSound (D6.x only)</td>
<td>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.</td>
</tr>
<tr>
<td>MixBufferMs</td>
<td>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.</td>
</tr>
<tr>
<td>MixIntPeriodMs</td>
<td>Interrupt interval, defaults to 200 milliseconds. Used only when MixServiceMode is 0.</td>
</tr>
<tr>
<td>MixIntResolutionMs</td>
<td>Interrupt duration, defaults to 50 milliseconds. Used only when MixServiceMode is 0.</td>
</tr>
<tr>
<td>SoundLevel0 through SoundLevel7</td>
<td>Specifies “wave output” volume corresponding to the soundLevel property. Default values for the soundLevel = 0 to 7 are shown: 1</td>
</tr>
<tr>
<td></td>
<td>SoundLevel0 = 0</td>
</tr>
<tr>
<td></td>
<td>SoundLevel1 = 24770</td>
</tr>
<tr>
<td></td>
<td>SoundLevel2 = 35030</td>
</tr>
<tr>
<td></td>
<td>SoundLevel3 = 42903</td>
</tr>
<tr>
<td></td>
<td>SoundLevel4 = 49540</td>
</tr>
<tr>
<td></td>
<td>SoundLevel5 = 55388</td>
</tr>
<tr>
<td></td>
<td>SoundLevel6 = 60674</td>
</tr>
<tr>
<td></td>
<td>SoundLevel7 = 65535</td>
</tr>
<tr>
<td>HighSpoolBufferMs</td>
<td>Length of one 16-bit spool buffer; default is 1500 milliseconds.</td>
</tr>
<tr>
<td>LowSpoolBufferMs</td>
<td>Length of one 8-bit spool buffer; default is 2500 milliseconds.</td>
</tr>
<tr>
<td>SpoolBufferAlloc</td>
<td>0: Allocates/deallocates spool buffers dynamically, when sound starts/stops (default) 1: Allocates spool buffer once at startup and keeps them it entire session</td>
</tr>
<tr>
<td>SpoolBufferCount</td>
<td>Number of spool buffers to allocate, from 2 (default) to 10.</td>
</tr>
</tbody>
</table>

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**

```lingo
on keyDown
    -- 0 through 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**

<table>
<thead>
<tr>
<th>Command</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>sound fadeIn channel {, ticks}</code></td>
<td>Fades volume up from 0 to the current volume of sound setting, not up to 255, as in D5. See the D6 ReadMe.</td>
</tr>
<tr>
<td><code>sound fadeOut channel {, ticks}</code></td>
<td>Fades volume down from the current volume of sound setting to 0, not from 255 to 0, as in D5. See the D6 ReadMe.</td>
</tr>
<tr>
<td><code>the sound of member videoMember</code></td>
<td>Boolean indicating whether sound is enabled for a digital video or Flash cast member.</td>
</tr>
<tr>
<td><code>the soundEnabled</code></td>
<td>If FALSE, all sounds are muted.</td>
</tr>
<tr>
<td><code>the soundLevel</code></td>
<td>System volume from 0 (mute) to 7 (loudest). See Table 15-11.</td>
</tr>
<tr>
<td><code>the visibility of sprite</code></td>
<td>Muting a Sound channel or sprite channel in the Score prevents the sound, SWA, or digital video sprite from playing.</td>
</tr>
<tr>
<td><code>the volume of member</code></td>
<td>Volume level for SWA members (0 to 255). Doesn’t work for Flash.</td>
</tr>
<tr>
<td><code>the volume of sound</code></td>
<td>Volume level for Sound channel 1 or 2 (0 to 255).</td>
</tr>
<tr>
<td><code>the volume of sprite</code></td>
<td>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.</td>
</tr>
<tr>
<td><code>the volumeLevel of sprite</code></td>
<td>Volume level for QT3 digital video sprites in D6.5. Use <code>volume of sprite</code> in D7. Nominal range is 0 to 255, but can be set much higher.</td>
</tr>
</tbody>
</table>

**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:


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.
Other Sound-Related Lingo

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.

<table>
<thead>
<tr>
<th>PowerPC</th>
<th>Mac 68K</th>
<th>Win 32</th>
<th>Win 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIX Services</td>
<td>MIX Services</td>
<td>mix32.X32</td>
<td>mix16.X16</td>
</tr>
<tr>
<td>Sound Import Export</td>
<td>Sound Import Export 68k</td>
<td>Sound Import Export.X32</td>
<td>mixsound.X16</td>
</tr>
<tr>
<td>SWA Streaming PPC Xtra</td>
<td>SWA Streaming 68K Xtra</td>
<td>swastrm.X32</td>
<td>swastrm.X16</td>
</tr>
</tbody>
</table>
Other Sound-Related Lingo

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.

---

**Table 15-13: Xtras Needed for Sound Playback. (continued)**

<table>
<thead>
<tr>
<th></th>
<th>PowerPC</th>
<th>Mac 68K</th>
<th>Win 32</th>
<th>Win 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>SWA Decompression PPC Xtra</td>
<td>SWA Decompression 68K Xtra</td>
<td>swadcmpr.X32</td>
<td>swadcmpr.X16</td>
<td></td>
</tr>
<tr>
<td>InetUrl PPC Xtra</td>
<td>None needed</td>
<td>InetUrl.X32</td>
<td>InetUrl.X16</td>
<td></td>
</tr>
<tr>
<td>NetLingo PPC Xtra</td>
<td>NetLingo 68K Xtra</td>
<td>NetLingo.X32</td>
<td>NetLingo.X16</td>
<td></td>
</tr>
<tr>
<td>NetManage WinSock Lib</td>
<td>None needed</td>
<td>None needed</td>
<td>None needed</td>
<td></td>
</tr>
<tr>
<td>MPEG 3 Import Export</td>
<td>MPEG 3 Import Export 68K</td>
<td>MPEG3 Import Export.X32</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>N/A</td>
<td>N/A</td>
<td>QT3Asset.X32</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>N/A</td>
<td>N/A</td>
<td>MacroMix.X32</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>N/A</td>
<td>N/A</td>
<td>DirectSound.X32</td>
<td>N/A</td>
<td></td>
</tr>
</tbody>
</table>

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/nutsbell/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.headspace.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

<table>
<thead>
<tr>
<th>Din Command</th>
<th>Description/Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>getChannelCount()</td>
<td>Returns maximum number of sound channels.</td>
</tr>
<tr>
<td></td>
<td>set numSoundChannels = getChannelCount()</td>
</tr>
<tr>
<td>getFreeChannel()</td>
<td>Returns number of highest free sound channel:</td>
</tr>
<tr>
<td></td>
<td>set highestFreeSoundChannel = getFreeChannel()</td>
</tr>
</tbody>
</table>
Other Sound-Related Lingo

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:

```plaintext
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.

### Table 15-14: The Unsupported Din Xtra (continued)

<table>
<thead>
<tr>
<th>Din Command</th>
<th>Description/Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>getPlayStatus()</code></td>
<td>Determines if the sound is playing in specified channel:</td>
</tr>
<tr>
<td></td>
<td><code>set soundPlaying = getPlayStatus (soundChan)</code></td>
</tr>
<tr>
<td></td>
<td>Buggy; seems to always return <code>TRUE</code>. Use <code>soundBusy()</code>.</td>
</tr>
<tr>
<td><code>getSndLength()</code></td>
<td>Returns sound stream’s total length in milliseconds:</td>
</tr>
<tr>
<td></td>
<td><code>set length = getSndLength (DinInstance)</code></td>
</tr>
<tr>
<td><code>getSndPosition()</code></td>
<td>Returns sound stream’s current time in milliseconds:</td>
</tr>
<tr>
<td></td>
<td><code>set currentTime = getSndPosition (DinInstance)</code></td>
</tr>
<tr>
<td><code>pauseRawSound()</code></td>
<td>Pauses stream played through <code>playRawSound</code>:</td>
</tr>
<tr>
<td></td>
<td><code>pauseRawSound (DinInstance)</code></td>
</tr>
<tr>
<td><code>playRaw()</code></td>
<td>Plays the list of files through raw sound as one stream. Returns sound channel:</td>
</tr>
<tr>
<td></td>
<td>`set soundChan = playRaw (DinInstance, ¬ [file1, file2, ...], &quot;aif&quot;</td>
</tr>
<tr>
<td><code>playRegular()</code></td>
<td>Plays sound through <code>playSound</code> mechanism. Returns sound channel:</td>
</tr>
<tr>
<td></td>
<td>`set soundChan = playRegular (DinInstance, ¬ soundFileName, &quot;aif&quot;</td>
</tr>
<tr>
<td><code>setSoundTime()</code></td>
<td>Sets stream to specified time (in milliseconds):</td>
</tr>
<tr>
<td></td>
<td><code>setSoundTime(DinInstance, time)</code></td>
</tr>
<tr>
<td><code>stopSound()</code></td>
<td>Stops stream played through <code>playSound</code> or <code>playRawSound</code>:</td>
</tr>
<tr>
<td></td>
<td><code>stopSound (DinInstance)</code></td>
</tr>
<tr>
<td><code>stopSoundInChan()</code></td>
<td>Stops sound in <code>soundChan</code>. Returns current time:</td>
</tr>
<tr>
<td></td>
<td><code>set currentTime = stopSoundInChan(soundChan)</code></td>
</tr>
</tbody>
</table>

**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:

```plaintext
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 Multi-session” 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 xccc?”**
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/

Sound & Cue Points
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
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/detools.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
Digital Video in Director 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:

\[
\text{(the width of member)} \times \text{(the height of member)} \times \text{(bits per pixel/8)} \times \text{frames per second} \times \text{duration in seconds}
\]
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://www.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:

\[
\text{Average Data Rate} = \frac{\text{Size of the External Data File in KB} \times \text{Float of the Duration of Member}}{\text{Digital Video Time Scale}}
\]

(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:

\[
\text{set the preLoadRAM} = 2 \times 1024 \times 1024 \quad \text{-- 2 MB}
\]

Or use a percentage of the freeBlock, such as:

\[
\text{set the preLoadRAM} = 0.5 \times \text{the freeBlock}
\]

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 (#quickTime-Media) 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 cannot 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

<table>
<thead>
<tr>
<th>Platform</th>
<th>Supported QT Versions in D6.5</th>
<th>D6.X <code>#digitalVideo</code> Members Require</th>
<th>D6.5 and D7 <code>#quickTimeMedia</code> Members Require</th>
</tr>
</thead>
<tbody>
<tr>
<td>Macintosh 68K</td>
<td>QT2 or QT3</td>
<td>QT2 or QT3</td>
<td>Never supported</td>
</tr>
<tr>
<td>PowerMac/G3</td>
<td>QT2 or QT3</td>
<td>QT2 or QT3</td>
<td>QT3 and QT3 Xtra</td>
</tr>
<tr>
<td>Windows 3.1</td>
<td>16-bit QTW2 only</td>
<td>QTW2</td>
<td>Never supported</td>
</tr>
<tr>
<td>Windows 95/98/NT</td>
<td>16-bit QTW2, 32-bit QTW2,</td>
<td>16-bit or 32-bit QTW2, QT3 and QT3 Xtra</td>
<td></td>
</tr>
<tr>
<td></td>
<td>QTW3 concurrently installed</td>
<td>matching Projector</td>
<td></td>
</tr>
<tr>
<td>Shockwave movie</td>
<td>QT2 (SW6); QT3 (SW7)</td>
<td>SW6¹</td>
<td>QT3 and QT3 Xtra¹</td>
</tr>
<tr>
<td>Browser outside</td>
<td>QT2 and QT3</td>
<td>QT2 browser plug-in</td>
<td>QT3 browser plug-in²</td>
</tr>
<tr>
<td>Shockwave</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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](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** if 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*

```lingo
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 systems are MCI-enabled, you can also use the Buddy API Xtra's `baVersion()` method:

```lisp
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 `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**

```lisp
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
```

```lisp
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
```
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:

```graphviz
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.)

---

Example 16-2: Checking for QT2 and QT3 Presence in D6.5 (continued)

```javascript
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
```

---

Digital Video in Director
Table 16-2 explains testing the `quickTimePresent` and `quickTimeVersion()` under Windows.

<table>
<thead>
<tr>
<th>Projector</th>
<th>16-bit QTW2</th>
<th>32-bit QTW2</th>
<th>QTW3</th>
<th><code>quickTimePresent</code></th>
<th><code>quickTimeVersion()</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>16-bit</td>
<td>N/A</td>
<td>N/A</td>
<td>FALSE</td>
<td>N/A</td>
<td>Error²</td>
</tr>
<tr>
<td>16-bit</td>
<td>✓</td>
<td>N/A</td>
<td>TRUE</td>
<td>N/A</td>
<td>Error</td>
</tr>
<tr>
<td>32-bit</td>
<td>N/A</td>
<td></td>
<td>FALSE</td>
<td>FALSE</td>
<td>0.0000</td>
</tr>
<tr>
<td>32-bit</td>
<td>N/A</td>
<td>✓</td>
<td>TRUE</td>
<td>FALSE</td>
<td>2.0000, 2.1200, etc.</td>
</tr>
<tr>
<td>32-bit</td>
<td>N/A</td>
<td>✓</td>
<td>FALSE</td>
<td>TRUE</td>
<td>3.0000, 3.0100, 3.0200</td>
</tr>
<tr>
<td>32-bit</td>
<td>N/A</td>
<td>✓✓</td>
<td>TRUE</td>
<td>TRUE</td>
<td>3.0000, 3.0100, 3.0200</td>
</tr>
</tbody>
</table>

1 The number of digits of the returned value displayed depends on the `floatPrecision`.
2 The `quickTimeVersion()` method reports 2.000 for any versions of QuickTime 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**

```lisp
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:

```lisp
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*

<table>
<thead>
<tr>
<th></th>
<th>Xtra Name</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>QuickTime Asset Options PPC (D6.5)</td>
</tr>
<tr>
<td></td>
<td>QuickTime Asset Options (D7)</td>
</tr>
<tr>
<td>Power Macintosh Authoring</td>
<td>QuickTime Asset PPC (D6.5)</td>
</tr>
<tr>
<td>Power Macintosh Distribution (Projectors or Shockwave)</td>
<td>QuickTime Asset (D7)</td>
</tr>
<tr>
<td>Windows 32-bit Authoring</td>
<td>QuickTime Asset Options.X32 (D6.5)</td>
</tr>
<tr>
<td>Windows 32-bit Distribution (Projectors or Shockwave)</td>
<td>QTAuth.X32 (D7)</td>
</tr>
<tr>
<td></td>
<td>QTExport.X32 (D7)</td>
</tr>
<tr>
<td></td>
<td>QuickTime Asset.X32 (D6.5)</td>
</tr>
<tr>
<td></td>
<td>QT3Asset.X32 (D7)</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Feature</th>
<th>Macintosh</th>
<th>Windows</th>
</tr>
</thead>
<tbody>
<tr>
<td>Video formats for import</td>
<td>QT2 (.MOV, &quot;MooV&quot;); D6.5 or later imports QT3, VFW (.AVI, &quot;VW&quot;),¹ and MPEG (.MPG).²</td>
<td>QTW2 (.MOV),³ VFW (.AVI), QT3 Xtra (D6.5) imports QTW3 formats, but not MPEG.</td>
</tr>
<tr>
<td>Video formats for export</td>
<td>QT2 (prior to D7); QT3 (D7 or later).</td>
<td>VFW (.AVI).</td>
</tr>
<tr>
<td>Standard video controller</td>
<td>Optional for QT2 and QT3.</td>
<td>Optional for QTW, not supported for AVI files.</td>
</tr>
<tr>
<td>QuickTime movie format</td>
<td>Allows file dependencies and resource fork.</td>
<td>QT2 movies must be flattened and de-forked. QT3 allows file dependencies.</td>
</tr>
<tr>
<td>QuickTime software</td>
<td>Extension is included with Mac OS, but can be disabled.</td>
<td>Must install QTW version to match Projector. (16-bit or 32-bit). D7 requires QTW3 (32-bit).</td>
</tr>
<tr>
<td>the timeScale of member property</td>
<td>Defaults to 600 for QuickTime.</td>
<td>Defaults to 30 for QTW and to 60 for VFW.</td>
</tr>
<tr>
<td>Multiple audio track support</td>
<td>Yes for QT2 and QT3.</td>
<td>No for AVI and QTW2; yes for QTW3.</td>
</tr>
<tr>
<td>Simultaneous Director and DV audio</td>
<td>Yes.</td>
<td>Not always. See “Sound Mixing under Windows” in Chapter 15.</td>
</tr>
<tr>
<td>Different custom palettes for multiple digital videos</td>
<td>QT2 requires Fix Palette Xobject to reset palette between movies. QT3 does not.</td>
<td>Does not require any special accommodations.</td>
</tr>
</tbody>
</table>

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 downloaded for free ([http://www.apple.com/quicktime](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](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:  


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:


To license the QT installer(s), download the PDF file:


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 seize the sole Windows sound channel when it plays a sound.

Whichever component takes control of the sound channel first may lock out the others 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.
Digital Video Tools and Options

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

<table>
<thead>
<tr>
<th>Action</th>
<th>Command</th>
</tr>
</thead>
</table>
| 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 ➤ Export²,³  
Format: QuickTime Movie (Mac: D6, D7; Windows: D7 only)  
Xtras ➤ QuickTime Sprite Export⁴ |

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.  

### 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.
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.
Digital Video Tools and Options

Stage display options

Table 16-7 summarizes the supported modes for direct-to-Stage video and the DV Controller.

Table 16-6: QT2 and QT3 Framing Options

<table>
<thead>
<tr>
<th>Framing</th>
<th>Center</th>
<th>Display</th>
<th>QT2</th>
<th>QT3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crop¹</td>
<td>TRUE</td>
<td>Center of video displayed; edges may be cropped.</td>
<td>Cropped, not scaled.</td>
<td>Scaled by scale of member, then cropped.</td>
</tr>
<tr>
<td>Crop¹</td>
<td>FALSE</td>
<td>Upper left of video displayed; lower right may be cropped.</td>
<td>Cropped, not scaled.</td>
<td>Scaled by scale of member, then cropped.</td>
</tr>
<tr>
<td>Scale²</td>
<td>N/A</td>
<td>Video stretched or shrunk to match sprite’s rect (no cropping)</td>
<td>Scaled based on size of sprite. No cropping.</td>
<td>Same as QT2. The scale of member is ignored.</td>
</tr>
</tbody>
</table>

¹ The crop of member is TRUE.
² The crop of member is FALSE.

Table 16-7: Support for Direct-to-Stage Video

<table>
<thead>
<tr>
<th>Platform</th>
<th>DV Format</th>
<th>Direct-to-Stage Setting</th>
<th>Controller</th>
<th>Ink Effects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Macintosh</td>
<td>QuickTime 2</td>
<td>True</td>
<td>Optional</td>
<td>Copy only</td>
</tr>
<tr>
<td>Macintosh</td>
<td>QuickTime 2</td>
<td>False</td>
<td>No</td>
<td>Limited¹</td>
</tr>
<tr>
<td>Windows</td>
<td>QuickTime 2</td>
<td>Always true</td>
<td>Optional</td>
<td>Copy only</td>
</tr>
<tr>
<td>Windows</td>
<td>QuickTime 3</td>
<td>True</td>
<td>Optional</td>
<td>Copy or custom mask only</td>
</tr>
<tr>
<td>Windows</td>
<td>VFW (AVI)</td>
<td>True</td>
<td>No</td>
<td>Copy only</td>
</tr>
<tr>
<td>Windows</td>
<td>VFW (AVI)</td>
<td>False</td>
<td>No</td>
<td>Limited¹</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Dialog Option</th>
<th>Lingo Equivalent</th>
<th>See Also</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cast Member Name</td>
<td>the name of member</td>
<td>castLibNum of member</td>
</tr>
<tr>
<td>File Name</td>
<td>the fileName of member (includes path)</td>
<td>name of member</td>
</tr>
<tr>
<td>Dimensions</td>
<td>the width, height, and rect of member</td>
<td>width, height, and rect of sprite</td>
</tr>
<tr>
<td>Duration</td>
<td>the duration of member / float (the digitalVideoTimeScale)</td>
<td>movieTime of sprite</td>
</tr>
<tr>
<td>Size</td>
<td>Size of header for QT2 and external file size for QT3. See Example 4-6.</td>
<td>size of member, duration of member</td>
</tr>
<tr>
<td>FrameRate/Frames Per Second</td>
<td>the frameRate of member. See “Determining a DV’s intrinsic frameRate.”</td>
<td>movieRate of sprite</td>
</tr>
<tr>
<td>Video</td>
<td>the video of member1</td>
<td>sound of member</td>
</tr>
<tr>
<td>Sound</td>
<td>the sound of member2</td>
<td>video of member, volume or volumeLevel of sprite, soundLevel, soundEnabled, trackEnabled, setTrackEnabled</td>
</tr>
</tbody>
</table>
Table 16-9 lists the properties for both #digitalVideo (QT2 and AVI) and #quickTimeMedia (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.

### Digital Video Properties and Functions

Table 16-9 lists the properties for both #digitalVideo (QT2 and AVI) and #quickTimeMedia (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.

<table>
<thead>
<tr>
<th>Dialog Option</th>
<th>Lingo Equivalent</th>
<th>See Also</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paused</td>
<td>the pausedAtStart of member (unrelated to the pause command)</td>
<td>controller of member, movieRate of sprite</td>
</tr>
<tr>
<td>Loop</td>
<td>the loop of member</td>
<td>movieTime of sprite, duration of member</td>
</tr>
<tr>
<td>Crop</td>
<td>the crop of member = TRUE</td>
<td>center of member, scale of member</td>
</tr>
<tr>
<td>Scale</td>
<td>the crop of member = FALSE</td>
<td>center of member, width, height, stretch, and rect of sprite</td>
</tr>
<tr>
<td>Center</td>
<td>the center of member</td>
<td>crop of member</td>
</tr>
<tr>
<td>Direct to Stage</td>
<td>the directToStage of member</td>
<td>ink, controller, invertMask, and mask of member</td>
</tr>
<tr>
<td>Controller</td>
<td>the controller of member</td>
<td>directToStage, pausedAtStart of member</td>
</tr>
<tr>
<td>Sync to Soundtrack</td>
<td>the frameRate of member = 0</td>
<td>movieRate of sprite</td>
</tr>
<tr>
<td><strong>Play Every Frame (No Sound)</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Normal</td>
<td>the frameRate of member = -1</td>
<td>movieRate of sprite</td>
</tr>
<tr>
<td>Maximum</td>
<td>the frameRate of member = -2</td>
<td>movieRate of sprite</td>
</tr>
<tr>
<td>Fixed fps</td>
<td>the frameRate of member = $n$ ($n &gt; 1$, specified in fps field)</td>
<td>fps field in dialog box</td>
</tr>
<tr>
<td>Enable Preload</td>
<td>the preLoad of member</td>
<td>the preLoadRAM, preLoadMember</td>
</tr>
<tr>
<td>Unload Settings (Normal/Next/Last/Never)</td>
<td>the purgePriority of member = $n$</td>
<td>unLoad, unLoadMember</td>
</tr>
</tbody>
</table>

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.
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:

```plaintext
set the movieRate of sprite 5 = 1
```

or in D7 syntax:

```plaintext
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

<table>
<thead>
<tr>
<th>Command</th>
<th>Value</th>
<th>Read-Only</th>
<th>Member</th>
<th>Sprite</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>bottom</td>
<td>Pixels from top of Stage.</td>
<td></td>
<td>✓</td>
<td></td>
<td>Includes height of controller, if any.</td>
</tr>
<tr>
<td>center</td>
<td>TRUE</td>
<td>FALSE</td>
<td>✓</td>
<td></td>
<td>Used only if crop = TRUE.</td>
</tr>
<tr>
<td>controller</td>
<td>TRUE</td>
<td>FALSE</td>
<td>✓</td>
<td></td>
<td>Used only if directToStage = TRUE.</td>
</tr>
<tr>
<td>crop</td>
<td>TRUE</td>
<td>FALSE</td>
<td>✓</td>
<td></td>
<td>See center, scale, translation entries. If crop is FALSE, movie is scaled to sprite box.</td>
</tr>
<tr>
<td>digitalVideoTimeScale</td>
<td>Default is 60 units/second.</td>
<td></td>
<td></td>
<td></td>
<td>A system property, not a sprite or member property.</td>
</tr>
<tr>
<td>digitalVideoType</td>
<td>#quickTime</td>
<td>#videoFor-</td>
<td>✓</td>
<td>✓</td>
<td>Always #quickTime for QT3 members.</td>
</tr>
<tr>
<td>directToStage(^1)</td>
<td>TRUE</td>
<td>FALSE</td>
<td>✓</td>
<td></td>
<td>See controller, mask, mouseLevel entries.</td>
</tr>
<tr>
<td>duration</td>
<td>Measured in ticks.</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>See mTime, movieTime entries.</td>
</tr>
<tr>
<td>fileName(^2,3)</td>
<td>Path to external video.</td>
<td>✓</td>
<td></td>
<td></td>
<td>Replace folder name in file path with @ in D6.5.</td>
</tr>
<tr>
<td>frameRate of member</td>
<td>0</td>
<td>-1</td>
<td>-2</td>
<td>n</td>
<td>✓</td>
</tr>
</tbody>
</table>
### Table 16-9: Digital Video and QuickTime 3 Properties (continued)

<table>
<thead>
<tr>
<th>Command</th>
<th>Value</th>
<th>Read-Only</th>
<th>Member</th>
<th>Sprite</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>height</td>
<td>Measured in pixels.</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>Should be multiple of 8.</td>
</tr>
<tr>
<td>ink</td>
<td>Default is 0 (Copy).</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>Copy ink is used for direct-to-Stage sprites.</td>
</tr>
<tr>
<td>invertMask4</td>
<td>TRUE</td>
<td>FALSE</td>
<td>✓</td>
<td></td>
<td>See mask entry.</td>
</tr>
<tr>
<td>isVRmovie4</td>
<td>TRUE</td>
<td>FALSE</td>
<td>✓</td>
<td>✓</td>
<td>✓ should be multiple of 8.</td>
</tr>
<tr>
<td>left</td>
<td>Pixels from left of Stage.</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>loaded</td>
<td>TRUE</td>
<td>FALSE</td>
<td>✓</td>
<td></td>
<td>Indicates only whether header is loaded, not data.</td>
</tr>
<tr>
<td>loop</td>
<td>TRUE</td>
<td>FALSE</td>
<td>✓</td>
<td></td>
<td>See loopBounds entry.</td>
</tr>
<tr>
<td>loopBounds4</td>
<td>[startTicks, endTicks]</td>
<td>✓</td>
<td></td>
<td></td>
<td>[0, 0] indicates no loopBounds.</td>
</tr>
<tr>
<td>mask4</td>
<td>member oneBitMember</td>
<td>✓</td>
<td></td>
<td></td>
<td>Set to 1-bit cast member or zero to disable; Only if directToStage = TRUE.</td>
</tr>
<tr>
<td>media</td>
<td>Not useful in most cases.</td>
<td>✓</td>
<td></td>
<td></td>
<td>Does not access external DV file data.</td>
</tr>
<tr>
<td>modified</td>
<td>TRUE</td>
<td>FALSE</td>
<td>✓</td>
<td></td>
<td>Indicates whether video has been edited in Video window.</td>
</tr>
<tr>
<td>mouseLevel4</td>
<td>#none</td>
<td>#controller</td>
<td>#all</td>
<td>#share</td>
<td>See VRTriggerCallback in Table 16-10.</td>
</tr>
<tr>
<td>movieRate6</td>
<td>0 – n, default is 1.0.</td>
<td>✓</td>
<td></td>
<td></td>
<td>Use mRate instead for QT3 sprites in SW6.0.1 and D6.5.</td>
</tr>
<tr>
<td>movieTime</td>
<td>Measured in ticks by default.</td>
<td>✓</td>
<td></td>
<td></td>
<td>Use mTime instead for QT3 sprites in SW6.0.1 and D6.5.</td>
</tr>
<tr>
<td>mRate4,6</td>
<td>0 – n, default is 1.0.</td>
<td>✓</td>
<td></td>
<td></td>
<td>See pausedAtStart; use for SW6.0.1 and D6.5 instead of movieRate.</td>
</tr>
<tr>
<td>mTime4</td>
<td>Measured in ticks.</td>
<td>✓</td>
<td></td>
<td></td>
<td>Use for SW6.0.1 and D6.5 instead of movieTime.</td>
</tr>
<tr>
<td>name</td>
<td>String.</td>
<td>✓</td>
<td></td>
<td></td>
<td>Defaults to initial external file name, excluding the path.</td>
</tr>
<tr>
<td>purgePriority</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3 (default is 3: Normal)</td>
<td>✓</td>
</tr>
</tbody>
</table>
Table 16-9: Digital Video and QuickTime 3 Properties (continued)

<table>
<thead>
<tr>
<th>Command</th>
<th>Value</th>
<th>Read-Only</th>
<th>Member</th>
<th>Sprite</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>rect</td>
<td>Measured in pixels.</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>Dimensions should be multiple of 8, and sprite rect should be even</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>multiple of member rect.</td>
</tr>
<tr>
<td>regPoint</td>
<td>point ((x, y))</td>
<td>✓</td>
<td></td>
<td></td>
<td>Defaults to center.</td>
</tr>
<tr>
<td>right</td>
<td>Pixels from left of Stage.</td>
<td>✓</td>
<td></td>
<td></td>
<td>Should be multiple of 8.</td>
</tr>
<tr>
<td>rotation</td>
<td>0.0 to 360.0 degrees.</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>Default is 0.</td>
</tr>
<tr>
<td>scale</td>
<td>([xScale, yScale])</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>Used if crop = TRUE. Default is ([100.0, 100.0]).</td>
</tr>
<tr>
<td>sound</td>
<td>TRUE | FALSE</td>
<td>✓</td>
<td></td>
<td></td>
<td>Ignored if frameRate is not 0 (sync to soundtrack).</td>
</tr>
<tr>
<td>startTime</td>
<td>Ticks.</td>
<td>✓</td>
<td></td>
<td></td>
<td>Unreliable.</td>
</tr>
<tr>
<td>stopTime</td>
<td>Ticks.</td>
<td>✓</td>
<td></td>
<td></td>
<td>Unreliable.</td>
</tr>
<tr>
<td>timeScale</td>
<td>Units/second.</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>Defaults to 600 for QT on Mac. See the digitalVideoTimeScale entry.</td>
</tr>
<tr>
<td>top</td>
<td>Pixels from top of Stage.</td>
<td>✓</td>
<td></td>
<td></td>
<td>Should be multiple of 8.</td>
</tr>
<tr>
<td>trails</td>
<td>TRUE | FALSE</td>
<td>✓</td>
<td></td>
<td></td>
<td>If directToStage = TRUE, DV leaves trails.</td>
</tr>
<tr>
<td>translation</td>
<td>([nPixels, vPixels])</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>Used if crop = TRUE.</td>
</tr>
<tr>
<td>type</td>
<td>#digitalVideo (QT2 or AVI) | #quickTimeMedia (QT3)</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>See digitalVideoType entry.</td>
</tr>
<tr>
<td>video</td>
<td>TRUE | FALSE</td>
<td>✓</td>
<td></td>
<td></td>
<td>See sound entry.</td>
</tr>
<tr>
<td>volume</td>
<td>0 to 255 (or higher)(^7)</td>
<td>✔️</td>
<td></td>
<td></td>
<td>Always reports 0 for QT3 in D6.5; see volumeLevel entry.</td>
</tr>
<tr>
<td>volumeLevel</td>
<td>0 to 255 (or higher)(^7)</td>
<td>✓</td>
<td></td>
<td></td>
<td>Use in D6.5 instead of volume.</td>
</tr>
<tr>
<td>width</td>
<td>Measured in pixels.</td>
<td>✓</td>
<td></td>
<td></td>
<td>Should be multiple of 8.</td>
</tr>
</tbody>
</table>

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 file name 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:

```plaintext
set the VRmotionQuality of sprite 5 = #minQuality
```

The VR prefix for each property is unnecessary in D7. For example:

```plaintext
sprite(5).motionQuality = #minQuality
```

### Table 16-10: QTVR-Specific Properties of QT3 Assets

<table>
<thead>
<tr>
<th>Command</th>
<th>Value</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>VRFieldOfView</td>
<td>degrees</td>
<td>Current field of view.</td>
</tr>
<tr>
<td>VRHotSpotEnterCallback</td>
<td>#enterHotspot</td>
<td>0 on enterHotspot me, hotSpotID actions</td>
</tr>
<tr>
<td>VRHotSpotExitCallback</td>
<td>#exitHotspot</td>
<td>0 on exitHotspot me, hotSpotID actions</td>
</tr>
<tr>
<td>VRMotionQuality</td>
<td>#minQuality</td>
<td>#maxQuality</td>
</tr>
<tr>
<td>VRMovedCallback</td>
<td>#movedVR</td>
<td>0 on movedVR me actions</td>
</tr>
<tr>
<td>VNode</td>
<td>nodeID</td>
<td>Current node ID being displayed.</td>
</tr>
<tr>
<td>VRNodeEnterCallback</td>
<td>#nodeEnter</td>
<td>0 on nodeEnter me, nodeID actions</td>
</tr>
<tr>
<td>VRNodeExitCallback</td>
<td>#nodeExit</td>
<td>0 on nodeExit me, oldNode, newNode actions return 0</td>
</tr>
<tr>
<td>VRNodeType</td>
<td>#panorama</td>
<td>#object</td>
</tr>
<tr>
<td>VRPan</td>
<td>degrees</td>
<td>See VRTilt entry.</td>
</tr>
<tr>
<td>VRStaticQuality</td>
<td>#minQuality</td>
<td>#maxQuality</td>
</tr>
<tr>
<td>VRTilt</td>
<td>degrees</td>
<td>See VRPan entry.</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Command</th>
<th>Value</th>
<th>Notes</th>
</tr>
</thead>
</table>
| 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 16-10: QTVR-Specific Properties of QT3 Assets (continued)
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

---

**Table 16-11: QuickTime 3 Asset Xtra Commands and Functions (continued)**

<table>
<thead>
<tr>
<th>Command</th>
<th>System</th>
<th>Member</th>
<th>Sprite</th>
</tr>
</thead>
<tbody>
<tr>
<td>useQuickTimeStreaming (TRUE</td>
<td>FALSE)1</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>VREnableHotSpot (sprite qtvrSprite, hotSpotID, TRUE</td>
<td>FALSE)3</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>VRGetHotSpotRect (sprite qtvrSprite, hotSpotID)3</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VRNudge (sprite qtvrSprite, #left</td>
<td>#upLeft</td>
<td>#up</td>
<td>#upRight</td>
</tr>
<tr>
<td>VRPToHotSpotID (sprite qtvrSprite, point (the mouseH, the mouseV))3</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VRswing (sprite qtvrSprite, pan, tilt, fieldOfView)3,4</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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.
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

<table>
<thead>
<tr>
<th>DV Speed</th>
<th>movieRate or mRate</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stopped</td>
<td>( n = 0 )</td>
<td>See pausedAtStart and controller of member in Table 16-9.</td>
</tr>
<tr>
<td>Normal speed</td>
<td>( n = 1 )</td>
<td>DV plays at intrinsic frame rate defined when it was created.</td>
</tr>
<tr>
<td>Reverse normal speed</td>
<td>( n = -1 )</td>
<td>Backward video playback is inefficient and may be jerky. Sounds play in reverse.¹</td>
</tr>
<tr>
<td>Fast forward</td>
<td>( n &gt; 1 )</td>
<td>Sound plays quickly and at a higher pitch. Video may be jerky at high multiples.</td>
</tr>
<tr>
<td>Fast reverse</td>
<td>( n &lt; -1 )</td>
<td>Sounds play quickly, at higher pitch, and in reverse. Video will be jerky.</td>
</tr>
<tr>
<td>Slow-motion forward</td>
<td>( 0 &lt; n &lt; 1.0 )</td>
<td>Sounds play slowly and at a lower pitch.</td>
</tr>
<tr>
<td>Slow-motion reverse</td>
<td>( -1.0 &lt; n &lt; 0 )</td>
<td>Video may be jerky. Sounds play slowly, and at a lower pitch.¹</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Playback Speed</th>
<th>frameRate</th>
<th>Video</th>
<th>Audio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synchronized</td>
<td>( f = 0 )</td>
<td>Drops frames, if needed, to achieve intrinsic rate, multiplied by movieRate of sprite. See Table 16-12.</td>
<td>Yes¹</td>
</tr>
<tr>
<td>Normal</td>
<td>( f = -1 )</td>
<td>Played at the DV file’s intrinsic frame rate without dropping frames.</td>
<td>None</td>
</tr>
<tr>
<td>Maximum</td>
<td>( f = -2 )</td>
<td>Played as fast as possible without dropping frames.</td>
<td>None</td>
</tr>
<tr>
<td>Fixed</td>
<td>( 1 &lt;= f &lt;= 255 )</td>
<td>Played at the specified frame rate without dropping frames.</td>
<td>None</td>
</tr>
</tbody>
</table>

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**

```lisp
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**

<table>
<thead>
<tr>
<th>Position Within DV</th>
<th>movieTime or mTime of sprite Equivalent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beginning of video</td>
<td>0</td>
</tr>
<tr>
<td>t ticks from the beginning</td>
<td>t</td>
</tr>
<tr>
<td>s seconds from the beginning</td>
<td>s * 60</td>
</tr>
</tbody>
</table>
| 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                 |
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

<table>
<thead>
<tr>
<th>Action</th>
<th>Command</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count the number of tracks in a DV sprite or cast member</td>
<td>trackCount (sprite y)</td>
<td>See trackType()</td>
</tr>
<tr>
<td>trackCount (member x)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Determine the type of data in a track</td>
<td>trackType (sprite y, t)</td>
<td>Returns #video, #sound, #text, #music, #sprite, #unknown, #chapter, or VOID if out of range.</td>
</tr>
<tr>
<td>trackType (member x, t)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Retrieve the text content of a text track</td>
<td>trackText (sprite y, t)</td>
<td>Applies only to text tracks.</td>
</tr>
<tr>
<td>Determine whether a track is enabled in a DV sprite</td>
<td>trackEnabled (sprite y, t)</td>
<td>See setTrackEnabled(), the video, and the sound of member.</td>
</tr>
<tr>
<td>Enable/disable a track in a DV sprite</td>
<td>setTrackEnabled (sprite y, t) = TRUE</td>
<td>See trackEnabled(), the video, and the sound of member.</td>
</tr>
<tr>
<td>FALSE</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Controlling Digital Video Playback

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**

<table>
<thead>
<tr>
<th>Action</th>
<th>Command</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Determine when a track starts within a DV sprite or cast member&lt;sup&gt;3&lt;/sup&gt;</td>
<td><code>trackStartTime (sprite y, t)</code>&lt;br&gt;<code>trackStartTime (member x, t)</code></td>
<td>See the startTime, movieTime, and mTime of sprite.</td>
</tr>
<tr>
<td>Determine when a track ends within a DV sprite or cast member&lt;sup&gt;3&lt;/sup&gt;</td>
<td><code>trackStopTime (sprite y, t)</code>&lt;br&gt;<code>trackStopTime (member x, t)</code></td>
<td>See the stopTime, movieTime, and mTime of sprite, the duration of member.</td>
</tr>
<tr>
<td>Enable/disable all video tracks in a cast member, and any sprites that reference it</td>
<td><code>set the video of member x = TRUE</code>&lt;br&gt;<code>set the video of member x = FALSE</code></td>
<td>See <code>setTrackEnabled(), trackEnabled(), sound of member.</code></td>
</tr>
<tr>
<td>Enable/disable all audio tracks in a cast member, and any sprites that reference it</td>
<td><code>set the sound of member x = TRUE</code>&lt;br&gt;<code>set the sound of member x = FALSE</code></td>
<td>See <code>setTrackEnabled(), trackEnabled(), video of member.</code></td>
</tr>
</tbody>
</table>

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.<br>2 Tracks are enabled/disabled on a per sprite basis, not a per cast member basis.<br>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.

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

```lingo
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

```lingo
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

<table>
<thead>
<tr>
<th>Action</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rewind a video</td>
<td>set the movieTime of sprite x = 0</td>
</tr>
<tr>
<td>Determine whether a video is currently</td>
<td>if the movieRate of sprite x &lt;&gt; 0 then...</td>
</tr>
<tr>
<td>playing</td>
<td></td>
</tr>
<tr>
<td>Determine whether a video file has</td>
<td>if the movieTime of sprite x &gt;= the duration of the member of sprite x</td>
</tr>
<tr>
<td>reached its end</td>
<td>then...</td>
</tr>
<tr>
<td>Wait until a specific cue point has been</td>
<td>See the Tempo channel’s Wait for Cue Point option or the mostRecentCue</td>
</tr>
<tr>
<td>reached in the DV file</td>
<td>Point of member and isPastCuePoint().</td>
</tr>
<tr>
<td>Wait until a specific time in seconds is</td>
<td>if the movieTime of sprite x &lt;= t * 60 then go the frame</td>
</tr>
<tr>
<td>reached in the DV file</td>
<td></td>
</tr>
<tr>
<td>Wait until a specific time in seconds is</td>
<td>repeat while the movieTime of sprite x &lt;= t * 60</td>
</tr>
<tr>
<td>reached in the DV file while in a repeat</td>
<td>updateStage</td>
</tr>
<tr>
<td>loop</td>
<td>end repeat</td>
</tr>
</tbody>
</table>

```

`Digital Video`
Digital Video Resources

There are many QT-related URLs of interest. Here are a select few. See http://www.zeusprod.com/nutsell/links.html for many more.

Information

Getting started with QuickTime 3:


QuickTime Gazette:


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):


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

Table 16-17: Common Digital Video Operations (continued)

<table>
<thead>
<tr>
<th>Action</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shut off all DV-related audio</td>
<td>set the movieRate of sprite x = 0</td>
</tr>
<tr>
<td>Shut off all non-DV related audio</td>
<td>repeat with x = 1 to 8</td>
</tr>
<tr>
<td></td>
<td>puppetSound x, 0</td>
</tr>
<tr>
<td></td>
<td>sound stop x</td>
</tr>
<tr>
<td></td>
<td>end repeat</td>
</tr>
</tbody>
</table>
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:


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/heise/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.
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 data 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 DDI, 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.
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).

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/nutsbell/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.thaiana.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/qtv 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 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 components. Using QT VR 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.rspace.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 Macro-media:QTVR Xtra folder on the D6 CD or printed in the Message windows using:

```lingo
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

<table>
<thead>
<tr>
<th>Command</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>QTVREnter (xtra &quot;QTVRXtra&quot;)</td>
<td>Call once to initialize the Xtra.</td>
</tr>
<tr>
<td>set instance = new (xtra &quot;QTVRXtra&quot;)</td>
<td>Create an instance.</td>
</tr>
<tr>
<td>put mMessageList (xtra &quot;QTVRXtra&quot;)</td>
<td>Display documentation.</td>
</tr>
<tr>
<td>QTVRExit (xtra &quot;QTVRXtra&quot;)</td>
<td>Close the Xtra.</td>
</tr>
<tr>
<td>QTVROpen(instance, &quot;vrMovie.mov&quot;, &quot;left,top,right,bottom&quot;, “visible”</td>
<td>Opens QTVR in specified rectangle; returns error string, or EMPTY for success.</td>
</tr>
</tbody>
</table>

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 file-Name 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>
<thead>
<tr>
<th>QD3D Member Properties</th>
<th>QD3D Sprite Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>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, fillColor, frame, interpolation, lightBrightness, lightColor, lightDirection, modelCenter, modelFile, modelSize, position, rotation, scale, shading, specularColor, specularCoeff, texture, textureType, type</td>
<td>autoRotate, autoRotateAngle, badge, buttonDistance, buttonMove, buttonRotate, buttonZoom, cameraAspectRatio, cameraDirection, cameraFOV, cameraHeight, cameraHither, cameraPosition, cameraInterest, cameraType, cameraUpVector, cameraWidth, cameraYon, controller, directManipulation, dragAndDrop, frame, directManipulation, draggingDrop, frame, frame, modelCenter, modelSize, position, rotation, scale</td>
</tr>
</tbody>
</table>
Index

Symbols and Numbers
@ character in Shockwave, 394
4-bit (16-color) palettes, 205
8-bit (256-color) palettes, 204
16-bit palettes, 411
32-bit palettes, 411
32-bit Windows differences, 201
3D Dreams, 580
“404 url not found” error, 344

A
aborts, preventing, 243
absolute frame rate, 72
accelerator keys (menu shortcuts), 36–42
(see also shortcuts)
activateWindow event, 189
active windows, 186, 189
Active XtraBase, 302
activeCastLib property, 131
activeWindow property, 186, 189
movies and, 173
window focus and, 190
ActiveX controls
importing, 114
Shockwave-related, 330, 334
Actor Control Xtra, 313
actorList property, 266
movies and, 173
performance, 289
Adaptec Toast, 248
adaptive palettes (see custom palettes)
Add button (Xtras option), 326
Add Default button (Xtras dialog), 305, 307
Add Default button (Xtras option), 326
Add ink (sprites), 448
Add Network button (Xtras dialog), 305, 307, 326
Add Pin ink (sprites), 448
addVertex( ), 405
Adjust option (updating Score references), 108
Adobe Photoshop
custom palette creation, 419
filter Xtras, 297, 298
importing CLUT files, 114
paint layers, 398
registration points, 150
Adobe Premiere, 572
After Frame One setting, 123
AfterShock utility, 339
agent Xtras, 310
Airbrush settings dialog box, 436
Airbrush tool (Paint window), 439
Alert( ), 483
alert command, 482
alert dialog boxes, 482
alertHook property, 483
  Director vs. Projectors, 241
  movies and, 171
Align window, 38
alignment, 148, 163–165
cast members, 150
coordinates, 151–163
  Stage- vs. monitor-relative, 154–157, 162
digital video, 556
alignment of member property, 381, 383
  Lingo access lacking, 142
All Macintosh Models option, 222
allocating memory, 253–255
disposing of objects, 266
errors, 121, 282
  (see also memory)
alpha channels, 447
alpha of member property, 383
Alphamania Xtra, 209, 302, 455
alphanumeric comparisons, 214
alphaThreshold of member property, 445, 447, 458
ALT attribute, 341
Alt key, 210
  (see also shortcuts)
America Online, Shockwave support for, 331
analyzing memory, 269–276
Animate in Background option, 225
animated GIFs, 113, 398, 400
animation
cursors, 465
film loops, 4, 59–61
optimizing, 61–63
performance, 292
Score for, 57–59
sprite transformation properties, 70
techniques for, 55–60
cast members, 59
antiAlias of member property, 293, 383, 402
Anti-Alias option (Rich Text Cast Member Properties), 379
anti-aliasing, 379, 450
  white background with, 62
antiAliasThreshold of member property, 293, 383
AnySaver utility, 250
AOL, Shockwave support for, 331
Apple menu, 51, 473
customizing, 29
AppleScript language, 52
applicationPath property, 212, 351
  Director vs. Projectors, 241
  movies and, 171
Arc tool (Paint window), 439
area of rectangles (example), 153
arrow key shortcuts, 47
Arrow pointer (cursor), 462
ASCII versions of text, 385
assets (see castLibs; cast members)
assetXtraInstalled( ) (example), 327
asynchronous video playback, 567
attaching castLibs at runtime, 106
attributes (see properties)
audio (see sound)
Audio Xtra, 529
Auto Coloring, performance and, 29
Auto Distort (Xtras menu), 299
Auto Filter (Xtras menu), 299
AUTOEXEC.BAT file, 50
auto-indent feature, avoiding, 36
AutoLaunch, AutoLaunch Pro, 250
automask of member property, 470
automatic cast member unloading, 268
AutoPlay option (Macintosh), 248
AutoPlay Test Configurator, 249
auto-puppeting, 14, 137
AutoRun feature (Windows), 50, 249
AUTORUN.INF file, 249
auto-starting Projectors, 248–249
autoTab of member property, 381, 383
available memory, checking for, 272–276
AVI files
audio tracks (see sound)
cue points, 512
cursors and, 466
AVI window, 38
backColor of member property, 381, 396, 453–454
color depth and, 417
backColor of sprite property, 15, 409, 453
color depth and, 417
background animation, 61
background color
anti-aliasing and, 62
color chips, 451
foreground color matching, 11
background loading, 353
Background Transparent ink (sprites), 448
backgroundColor property, 402
backups, external castLibs and, 103
backwards looping, 4
baCommandArgs( ), 232
baCpuInfo( ), 231
baDisableScreenSaver( ), 232
baDisableSwitching( ), 232
baDiskInfo( ), 232
baGetVolume( ), 526
baHideTaskBar( ), 232
baMemoryInfo( ), 269
baMsgBox( ), 483
bandwidth, 63, 255, 286
(see also memory)
baPrevious( ), 232
BarbaBatch utility, 536
baScreenInfo( ), 231
baScreenSaverTime( ), 232
baSetDisplay( ), 231
baSetScreenSaver( ), 232
baSetVolume( ), 526
baSysFolder( ), 232
baVersion( ), 231, 545, 546
Beatnik Lite Xtra, 314, 485
Beatnik Xtra, 507, 529
MIDI sounds, 531
sound mixing, 502
beep( ), 522
beepOn property, 522
Before Frame One setting, 123
beginRecording command, 94
beginSprite event, 7–9, 133, 465
Behavior Inspector window, 38
Behaviors
external castLibs to Stage, 103
Internet Streaming Behaviors, 348
Lingo Behavior Database, xxii
beta versions, testing, 287
BGCOLOR attribute, 341
bgColor of member property, 383, 396, 453, 454
color depth and, 417
bgColor of sprite property, 409, 453
color depth and, 417
bgColor of the stage property, 168
bit depth, 19, 225, 265, 408
animation, 62
choosing, 411
converting to higher depths, 428
determining and changing, 233
palettes and, 5
performance and, 292
sound, 485
bitmap sprites (see sprites)
bitmaps, 398–400
alignment regPoints of, 164
buttons, 458
cursors, 463
external, performance and, 290
first animated GIF frames as, 400
importing, 113
internationalization concerns, 216
palette mapping/transform, 423
registration points, 121, 148, 150
rich text stored as, 259
size of, 257, 442, 450
bitmapSizes of member property, 388
bitRate of member property, 520
bits per sample (sound), 485
bitsPerSample of member property, 520
bkMixer Xtra, 526
Black Arrow (cursor), 462
Blank (hidden) cursor, 462
Blend ink effect, 446, 448
performance and, 292
blend of sprite property, 11, 15, 449
blendLevel of sprite property, 445, 449
blendSprite( ) (example), 449
boot managers, 27
border of member property, 381
Border Xtra, 179
borders, MIAWs, 180
bottom of rect property, 151, 161
bottom of sprite property, 148, 158, 561
bottomSpacing of member property, 383
bounding box, checking if on Stage, 157
boxDropShadow of member property, 381
boxType of member property, 381
bracket syntax (Director 7), xviii
BRAND.BRD file, 219
BreakConnection( ), 362
broadcasting messages, 289
broadcastProps property, 402
browser scripting, 356–359
browserName( ), 231, 346, 351, 354
browsers
  obtaining, 334
  palettes for, 423
  Shockwave and, 339–342
    communications between, 356–359
    memory requirements, 332
    support for, 345–348
Buddy API Xtra, 178, 180, 231, 237,
  243, 250, 251, 303, 373, 470
bzVersion( ), 545, 546
custom dialog boxes, 483
volume control, 526
budgetting memory, 263–267
buffers
  audio buffers, 255
  frame buffers, 9
  offscreen buffer, 9, 265, 271
bugs, 284
  addVertex( ) (D7.0), 406
  Apple menu option (D7.0), 473
  color depth (D4.0.4), 415
  counting lines/paragraphs (D7.0),
    386
  font settings (D6.0), 372
  fontStyle property (7.0), 387
  installMenu command (D7.0), 471,
    476
MIAWs (Director 7), 166, 191
preloading (D7.0), 278
Save as Java feature (D7), 339
starting Projector in D7.0, 251
traceLoad property (D7.0), 274
unLoadMember (D7.0), 268
build options, Projectors, 224
buildCastmemberNamesList( ) example, 145
built-in palettes, 421
buildling Xtras into Projectors, 203, 226,
  324–326
Button Editor Xtra, 461
buttons, 458–461
  editing sprites on Stage, 378
  interactivity for (example), 144
  storage space, 258
  types of, 459–461
  window titlebar, 180
buttons (mouse), 209
buttonStyle of member property, 460
buttonType of member property, 460
buying Xtras, 303–305
C
  cacheDocVerify( ), 351, 354
  cacheSize( ), 272, 351, 354
  caching
    cast member names, 130
    Xtras, 318
  cancelIdleLoad command, 280
  canUseQuicktimeStreaming( ), 565
capitalization
  conventions for, xiii
  platform differences, 210, 214
Cast
  Lingo with, 16
  loading, 265
  performance issues, 291
  Cast and Score (Memory Inspector), 271
  Cast Control Xtra, 313
  Cast Effects Xtra, 454
  Cast FX Xtra, 408
cast keyword, 102
Cast Member Properties dialog boxes,
  137, 379–380
cast members, 8–9, 123–129
  alignment, 150
  animating with, 59
  changing, sprites and, 8
  checking color depth, 412
  coordinates for, 158
  copying, 87, 128
  corrupted files and, 129
counting, 147
creating dynamically, 133, 284
as cursors, 463
deleting, 124, 127
  Score references and, 129
digital video (see digital video)
DV (see digital video)
film loops and, 60
header information, 19, 256
importing linked, 146
internal (unlinked/embedded), 102
libraries of (see castLibs)
Lingo for, 129–147
linking, 102
  at runtime, 123–124
  finding linked members, 145
loading and unloading, 19, 123, 267–282
  determining what is loaded, 275
  displaying loading/unloading, 285
  performance and, 55
  preloading manually, 276–280
modifying at runtime, 106
moving, 127
multiple castLibs, working with, 31
names for, 7, 16, 105, 130–132, 289
  caching, 130
  duplicates, searching for, 144
  numbers for, 6, 289
palettes for, 421
pop-up menus, 479–482
properties, 133–143
  changing, 145
  complete list of, 138–141
  idiosyncracies, 142
  setting, 137
purging, 267
resizing, 164
scripts with (see cast scripts), 8
sorting and searching for, 124
sound, 486–487, 489–491, 509
  (see also sound)
storage of, 252–253, 259, 278, 292
  loading and, 9
  order, 104
thumbnails, 66
transition cast members, 97
troubleshooting errors with, 128
  unused, 124
  X instead of, troubleshooting, 11
cast of member property, 136
cast of sprite property, 136
cast scripts, 8
Cast window, 101
castLib preferences, 111
  context-sensitive menu, 39
  moving cast members, 127
  opening multiple, 44, 102, 108
  shortcuts, 125–127
CastControl Xtra, 106
CastEffects Xtra, 106, 302
castLib command, 131
castLibNum of member property, 131
castLibNum of sprite property, 137
castLibs (cast libraries), 101–111
  adding to Xtras menu, 299
  Creator Codes, 32, 250
  external, 31, 102–108, 128
    memory leak with Sprite Xtras, 284
    protecting, 230
  Score references to, adjusting, 107
  unlinking and relinking, 105
  uses and disadvantages, 103–105
formats for, 109–110
importing (to) (see importing)
internal, 102
Lingo for, 129–147
listing, 132
loading, 19, 123
  performance and, 55
  storage space and, 9
management commands, 108
multiple, working with, 31
preferences, 111
Shared Cast vs., 106–107
storage, 252, 292
unlinking, 128
unneeded, 292
  (see also cast members)
castMemberList of member property, 470
castNum of sprite property, 135–137
castType of member property, 142
CCT files, 109
CD Pro Xtra, 245, 246, 533
CD-ROM drive
- access, performance of, 290
- customizing icons, 246–248
- data throughput, 262
- digital video performance from, 577
- disk storage capacity, 263
- Enhanced CDs (ECDs), 531–533
- locating for user, 244–246

cells
- Director 6/7 changes, 68
- duplicating, 67

center of member property, 560, 561
- Center option, 225
- center property, DV sprites, 142
- centering (see alignment)
- centerRegPoint property, 402
- centerStage property, 19, 162, 169, 351
- changeArea of member property, 73

Channel parameter (Tempo dialog), 512
- channelCount of member property, 509
- channels
  - Effects (see Effects channels)
  - examining all, 99–100
  - sound (see sound)
  - sprite (see sprite channels)

Character Map utility, 376
- character mapping, 375
- characterSet of member property, 388
- charPosToLoc( ), 389
- charSpacing of member property, 383
- check boxes, importing, 115

“Check Movie for Xtras” option, 305, 306, 322, 226
- checkBoxAccess property, 460
- checkboxes, 459–461
- checkBoxType property, 460
- checkDepth( ) (example), 412
- checkLinks( ) (example), 145
- checkMark of menuItem property, 477
- checkmarking menu items, 475
- CheckNetMessages( ), 362
- checkQTversions( ) (example), 544
- checkScore( ) (example), 78
- checkSeekOffset( ) (example), 275
- chunk expressions (text), 385–387
- chunkSize of member property, 73

Cinemac utility, 250

Classic Look (Monochrome) option, 427
- CLASSID attribute, 341
- clearCache( ), 351, 354
- clearFrame command, 95
- clearGlobals command, 171, 266
- clickLoc property, 159
- clickM( ), clickV( ) (examples), 160
- clickOn property, 143
- Clipboard ink effect, 446
- close box (windows), 180
- close window command, 188, 189, 192
- closed property, 402
- closeDA command, 212
- closeHandler event, 180
- closeResFile( ), 212, 468
- closeWindow event, 189
- closeXlib( ), 212, 266, 314
- closing external files, 284
- closing Stage window (D7), 42
- closing windows, 42–44, 187
- CLUT files, importing, 114
- Cmd key (see shortcuts)
- CMYK color model, 408
- CODEBASE attribute, 341
- codecs, 539
- collaboration, 103, 105
- color chips, 451–454
- color depth, 19, 206, 225, 265, 408
- animation, 62
- choosing, 411
- converting to higher depths, 428
- determining and changing, 233
- palettes and, 5
- performance and, 292

Color Genie Xtra, 455
- color models, 408–410
- color of member property, 383, 396, 453, 454
- color depth and, 417
- color of sprite property, 409, 453
- color depth and, 417

Color Palettes window, 39, 433
Color Picker Xtra, 455
- colorDepth property, 206, 231, 351, 396, 412
- fullColorPermit property and, 265
- multiple monitors and, 233
- troubleshooting, 416–417
Index 589

colorized text
  color text, 452
  performance and, 36
colorQD property, 396
colors
  background, anti-aliasing and, 62
  bitmap compression and, 257
desktop, Projectors and, 228
direct color modes, 411
Director 7 color picker, 409
inks (see ink effects)
networking concerns, 216
Lingo commands for, 207, 396–397
listing for cast members, 125
Paint window (see Paint window)
Palette channel, 5
palettes, 417–430
  conflicts, 424
  custom, 121
  importing, 113, 419–420
  interface options, 429
  Lingo properties, 432
  MIAWs and, 192
  operations on, 423
  reserving colors in, 425
  storage space for, 259
  types of, 420–423
platform differences, 204
reserved, 62, 204
same foreground and background, 11
transparency
  palettes and, 420
  Stage window (simulating), 182
troubleshooting, 425–430
Xtras for, 454
  Command key (Macintosh), 210
  command shortcuts, 35
  commandDown property, 36, 210
  CommandLine Xtra, 251
commands
  contained in Xtras, listing, 324
  menu items, 474
comments
  frame markers, 74, 76
  performance and, 290
CommPort XObject Xtra, 314
compacting (see compression)
comparing strings, performance, 289
compatibility testing, 31–32
Composite Xtra, 385
  “Compress (Shockwave Format)”
  option, 226
compressed format (movies, castLibs), 109
compressed option (Project creation), 227
compression, 252
  bitmaps, 257, 398
  determining compressed size, 275
  digital videos, 260, 538
  external movies, 230
  performance and, 290
  streaming media data rate and, 257
  SWA (Shockwave audio), 208, 260, 486, 516–520
CONFIG.SYS file, 50
configuring PopMenu cast members, 480
configuring system, 29–33
  cross-platform development, 198
  internationalization and, 215
  conflicts between palettes, 424
  ConnectToNetServer( ), 362
  constrainH( ), 161
  constrainV( ), 161
  context-sensitive menus, 35, 37–39, 49
  platform differences, 199
  Score window, 80
  Stage and sprites, 81
continuation character, xiv
Control key, 210
Control menu, 42
Control Panel (Windows), 38, 49, 87
communicator of member property, 560, 561
communicator property, DV sprites, 142
  converting data structures, 214, 289
  convertLabelsToList( ) (example), 76–78
Cool Edit 96 utility, 536
  creating cue points, 513
coordinates, 148, 151–163
  converting between systems of, 154
  properties of, 158–160
  Stage- vs. monitor-relative, 154–157, 162
  in user interface, 163–165
Copy ink effect (sprites), 447
Index

copying  
  Behaviors between castLibs, 103  
  cast members, 87, 128  
  cell duplication, 67  
  frames, 87–88, 96  
  cast members and, 128  
  markers, 75  
  sprites, 87, 128  
(see also pasting)
copyright symbol, 376
copyrightInfo of member property, 520
copyToClipboard( ), 351
corrupted  
  colors in Windows, 425  
  files, 129  
  Score, 291
cost of Xtras, 303–305
  “Could not display plug-in for MIME type Text/HTML” error, 343
count property, 383, 386
counting cast members, 147
countMembers( ) (example), 147
countVisibleMIAWs( ) (example), 187
cpuHogTicks property, 204, 289, 293
cpuIdGetCPUType( ), 231
cpuIdIsGenuineIntel( ), 231
CPUs, information on, 237
Create Projector dialog box, 222
cREATEFILE command, 212
Creator Codes, 32, 250
crop( ), 445
crop of member property, 560, 561
crop or member property  
  DV sprites, 142
cropping  
  digital video, 556
  MIAWs, 183
  in Paint window, 441
Crossbar cursor, 462
Crosshair cursor, 462
cross-platform differences, 199–216
  browser scripting support, 357
cursors, 211
date and time presentation, 215
determining platform, 234–238
filenames, 211–213
graphics file formats, 207
mapping fonts, 373
menus, 472
porting between platforms, 198
Projectors, 221–224
Shockwave plug-in support, 331
shutDown, restart commands, 229
sound, 486–487
strategy for, 195–199
system palettes, 423
text selection, 390
cross-platform Xtras, 304
CST files, 109
Ctrl key (see shortcuts)
Cue Card Xtra, 313
Cue Point option, 512
cue points, 208, 507, 510–516
  digital video, 538, 571
  events, 515
  Lingo for, 514
  waiting for, 72
cuePassed events, 515, 516
cuePointNames of member property, 514, 520
cuePointTimes of member property, 514, 520
currentSpriteNum property, 137, 143
currentTime of sprite property, 520
cursor command, 464, 468, 470
  movies and, 173
cursor of sprite property, 464, 465, 467, 468, 470
cursors, 462–471
  cast members as, 463
  coordinates, 159
  custom, 465, 468–470
  importing, 114
  Lingo for, 468
  operations on, 466
  platform differences, 211
  position, controlling, 470
  switching, memory leak when (D7.0), 284
cursorSize of member property, 470
custom  
  Apple menu configuration, 29
  palettes, 418–420
  user interface elements
    buttons, 258, 461
    cursors, 465, 468–470
    dialog boxes, 483
hierarchical and pop-up menus, 478–482
menus, 471–475
Xtras, 301
Custom Button cast members, 258
Custom Button Editor Xtra, 60
Custom Button Xtra, 258
Custom Cursor Xtra, 60, 211, 465, 468–470
Custom tile settings dialog box, 436
cut-and-paste, 394
CXT files, 109
Cycle ink effect, 446

delay (see latency)
delay command, 71, 92
deleteFrame command, 95
deleteVertex(), 406
deleting
automatic sprite erasure, 10
cast members, 124, 127
Score references and, 129
castLibs, 292
disposing of and purging items, 266
erasing in Paint window, 441
external castLibs, 107
frames, 87–88, 96
markers, 75
puppet sprites, 267
sprites before next marker, 76
windows, 187
delivery testing, 26–27
depth of member property, 207, 396, 398, 399, 413, 417, 468
desktop, hiding/showing with
Projectors, 228
desktop database (Macintosh), 52
desktop folder (Macintosh), 52
deskTopRectList property, 162, 168, 231, 232
destination chip in Paint window, 451
detaching castLibs at runtime, 106
detecting (see searching)
development environment, 25–26
diacritical marks, 214
dialog boxes, 482–484
Lingo for, 483
MIAWs for, 170
MUI Dialog Xtra, 313
DialogBoxer utility, 483
difference (tweened) frames, 4
editing sprite paths, 81–85
number of, 62
Digital Cafe, 250
digital video, 398, 537–582
compression and decompression, 538
cross-platform issues, 548
data rate, 256, 260, 279, 486
detecting software for, 543–546
editing, 554
formats other than QuickTime, 578–579
digital video (continued)
importing, 113
Lingo for, 569–570
not appearing on Stage, 11
palettes for, 422
performance, 290, 291, 539–541
platform dependencies, 208
playback control, 566–572
preloading, 279
properties and options, 554–566
file information, 554
framing options, 556
intrinsic frameRate value, 568
masks, 566
memory management, 559
playback options, 555
property idiosyncracies, 142
purging, 267
QTVR and VRML, 579
QuickDraw 3D Xtra, 581
registration points, 148, 150
resources for, 572–573
sound, 494
sound issues, 551
speed and synchronization, 539, 558, 567
Stage display options, 557
storage space for, 259
track control, 569–570
transitions and, 73, 207
troubleshooting, 573–578
user interface and, 551–566
Xtras for, 573
(see also direct-to-Stage sprites)
Digital Video Cast Member Properties
dialog box, 554–566
“Digital video sprite expected” error, 574
Digital Video window, 551, 554
button shortcuts, 45
context-sensitive menu, 38
opening multiple, 44
digitalVideoTimeScale property, 555, 561, 568, 570
digitalVideoType property, 561
Din Xtra, 529
DIR files, 32, 109
dirapi.mch file, 318
DIRDIB.DRV video driver, 18
direct color modes, 411
Direct Sound device, 497
“Direct to Stage” option, 380, 557
DIRECT-L mailing list, xxii
DirectMedia Xtra, 539, 573
Director
advice on using (see tips and tricks)
files (see files, Director)
hardware/software requirements, 24–29
Help system and manuals, 33–34
how it works, 3–5, 18–20
managing multiple files, 31
memory budget, 263–267
platforms and operating systems, 195–216
cross-platform strategy, 195–199
differences between, 199–216
(see also cross-platform differences; operating systems)
Projectors differences from, 238–243
resources for further reading, xx–xxiii
running multiple copies, 31
Shockwave vs., 349–353
sound playback (see sound)
system configuration, 29–33, 198
versions of
Creator Codes, 32
“Director 5 Style Score display” option, 65
Director 7 bugs (see bugs)
features of Director 7, xvi–xx
fonts in Director 7, 387
Score improvements, 63–69
Shared Cast, 106–107
Shockwave features and, 361–366
upgrading, 32
working with multiple, 31–33
(see also Macintosh; Windows operating systems)
Director 6.0 Xtra Cache file, 318
Director 7.0 Preferences file, 318
DIRECTOR.INI file, 18, 200
audio buffer settings, 523
swap space settings, 255
DirectOS Xtra, 250, 303, 373, 470
DirectSound sound driver, 501
DirectSound sound mixer, 502, 505–507
DirectSound Xtra, 529
DirectSound.X32 Xtra, 505
directToStage of member property, 400, 402, 560, 561, 574
digital video sprites, 142
direct-to-Stage sprites, when rendered, 10
DirMMX Xtra, 237, 292, 427
DirSaver utility, 250
disabled menu items, 476
disk space
digital video, 538, 555
disk space (see storage space)
disks
 access, performance of, 290
 asset size on, 256
 floppy, loading onto, 294
 storage capacity, 263
 swap space, 204
 Display menu, 67
 DisplayRes command, 231
 disposing of objects, 266
dissolves, 73
 (see also transitions)
distance between points (example), 154
distorting (see animation)
distributing Xtras, 322–324
distribution (see Projectors)
dither of member property, 424, 428, 445
dither property, 293
dithering, 423
DLL files, 50
DLLGLUE.DLL library, 314
DLLname command, 523
DLLs for Xtras, 317
DM Transitions Xtra, 73
do command, 289, 351
document files, starting Projectors with, 250
Don’t Adjust option, 108
Don’t Fade option, 430
DonationWare, 302
doneParsing( ), 366
DOS prompt (Windows), 50
dot syntax
 performance, 290
text member strings and, 386
dot syntax (Director 7), xviii
DOUG (Director Online Users Group), xxii
Dowdell, John, 174, 522
Download if Needed option (Xtras dialog), 307, 326
“Download n Frames Before” option, 348
downloading, external castLibs and, 103
downloadNetThing command, 277, 351, 354
URLs as source movies, 177
downsampling, 485
drag-and-drop
 Director 6/7 changes, 68
 importing media, 112
dragging sprites (example), 160
drawing
 film loops, 61
 frames to Stage, 9–11
 MIAWs, 175
 at runtime, 407
 (see also alignment; position on Stage)
drawRect of the stage property, 162
drawRect of window property, 162, 183–186
 panning and zooming MIAWs, 185
 redrawing MIAWs, 175
DrawXtra Xtra, 408
drive, CD-ROM, 244–246
drivesCD( ), 245
drivers, 50
drivesToList( ), 245
DropDownList Behavior, 478
dropShadow of member property, 381
DropStart utility, 248
DSWMEDIA folder, 344
dual monitor support, 24, 206
dual monitor support, 24, 206
colorDepth property, 235
coordinates and, 154
deskTopRectList property, 232
platform differences, 206
“Duplicate Cast Members for Faster Loading” option, 226
duplicate command, 131
DuplicateFrame command, 95
duplicating (see copying)
duration of member property, 73, 495, 520, 555, 559, 561
cue points and, 514
duration of sprite property, 561
DV (see digital video)
DV Cast Member Property dialog box, 567
DVD-ROM storage medium, 579
DVD-Video specification, 579
DXR files, 109
dynamic activity (see runtime)

**E**
EDCCTRL XObject, 246
ECDs (Enhanced CDs), 531–533
Edit menu accelerators, 40
editable of member property, 381, 391
editable of sprite property, 380, 381, 390
Editable option (Field Properties), 379
editable property, 15
editing sound, 535
Effector Set Xtra, 106, 302, 455
Effects channels, 5–6, 69–78
context-sensitive menus, 80
controlling with Lingo, 93
examining all, 99–100
hiding/showing, 66
Palette channel, 5
in scoreSelection list, 97
Sound channel, 6
Tempo channel, 5, 71–72
Transition channel, 6, 72–74
effects tools (Paint window), 443
8-bit (256-color) palettes, 204
email lists on Director, xxii
<EMBED> tags, 340–342
embedding
cast members, 102
fonts (Director 7), 387
sounds, 490
  compressed sounds, 517
“Empty at End” option, 124
Enable Preload option, 279, 559
enabled of menuItem property, 476, 477
encryption Xtra (rumored), 314
End key, 47
dendColor property, 402
dendRecording command, 94
dendSprite event, 7, 8, 465
Enhanced CDs (ECDs), 531–533
enterFrame event, 6, 10, 94
delay command and, 92
frame scripts and, 8
environment property, 231, 233
Equilibrium’s deBabelizer, 573
Equilibrium’s deBabelizer, 455
erase member command, 130, 131
Eraser tool (Paint window), 438
erasing in Paint window, 441
erasing sprites automatically, 10
errors
  checking during runtime, 233
  error codes, 215
  memory (see insufficient memory)
  memory errors, 282–286
  network error codes, 359–361
Ethernet cards and transceivers, 28
EvalScript event, 359
EvalScript( ), 358
event handlers
  changing sprite properties, 143
  external castLibs and, 105
  stylistic handling of on, xiii
events, 4
cue points, 515
delay command and, 92
Director 6/7 changes, 66
idle loading, 280–282
Lingo interference with, 14
MIAW anomalies, 191
MIAW messages, 188–190
Tempo frame rate and, 5
user events and sprite properties, 143–144
waiting for user events, 71, 93
examineList( ) (example), 77, 144
examples in this book, about, xiv–xvi
EXE Screen Saver utility, 250
exitFrame event, 10, 92
delay command and, 92
frame scripts and, 8
exitLock property, 173, 174, 209
explicitly unloading cast members, 268
Export Xtras, 298
exporting, 122–123
sound files, 486
Extended display mode, 67
Extensions folder (Macintosh), 52
external castLibs, 31, 102–108
copying cast members, 128
listing, 132
management commands, 108
maximum number of, 104
memory leak with Sprite Xtras, 284
protecting, 230
Score references to, adjusting, 107
Shared Cast vs., 106–107
storage, 253, 292
unlinking and relinking, 105
uses and disadvantages, 103–105
external files
bitmaps, performance and, 290
determining size, 146
palettes with, 422
platform differences, 211–213
remember to close, 284
external links (see linking)
external movies, 230
externalEvent( ), 351, 354, 357, 358
externalParamCount( ), 357
externalParamName( ), 357
externalParamValue( ), 358
ExtraMemory option, 255
Eyedropper tool (Paint window), 439

F
fading
palettes, 205, 431
sound, 524–526
Fast Eddy utility, 455
Field Cast Member Properties dialog box, 379
Field window
button shortcuts, 45
context-sensitive menu, 38
opening multiple, 44
fields, 369–376
advantages and disadvantages, 370
appearance and attributes, 370–376
attributes for, 379
fonts and formatting, 372–376
importing, 115
Lingo for, 380–389
storage space, 258
troubleshooting, 393
File menu accelerators, 39
file selection dialog boxes, 484
fileExists( ), 245
FileFlex Xtra, 302
FileIO Xtra, 102, 124, 484
creating Creator Codes, 251
performance issues, 290
platform differences, 212
fileName of castLib property, 131, 212, 351
replacing external castLibs, 105
fileName of member property, 212, 351, 400, 409, 559, 561
checking for available space, 272
dynamic linking to cast members, 123–124
dynamically accessing external assets, 102
“Where is...?” dialog box, 120
fileName of window property, 177, 212
filenames, platform differences, 211–213
FileOpenDialog( ), 484
files
corrupted, 129
digital video, information on, 554
Director files
multiple, working with, 31
transferring across networks, 28
external
bitmaps, performance and, 290
determining file size, 146
platform differences, 211–213
remembering to close, 284
graphics file formats, 397–408
platform dependencies, 207
importing, 112–115, 117–121
troubleshooting, 120–121
Macintosh file types, 52
Macintosh management, 53
media file formats
adding to movies, commands for, 122
for cast member properties, 138–141
files (continued)
media file formats (continued)
linking, advantages/disadvantages, 118
movies and castLibs, 109–110
platform dependencies, 207–209
Projectors, 226
sound file formats, 486
for sprite properties, 138–141
storage space and, 255–262
type of member property, 115–121
starting Projectors with documents, 250
storage (see compression; storage space)
text, linking to, 124
Windows OS file types, 49
FileSaveAsDialog( ), 484
FileXtra, 245, 302, 484
Fill tool (Paint window), 444
fillColor property, 402
fillCycles property, 402
fillDirection property, 402
fillDirection property, 402
filled of member property, 397
filled oval tool (Paint window), 440
filled polygon tool (Paint window), 440
filled rectangle tool (Paint window), 439
fillMode property, 403
fillOffset property, 403
fillScale property, 403
film loops, 59–61
backwards looping, 4
loping in first or last frames, 4
MIAWs to manipulate, 170
storage space, 259
Filter Bitmap (Xtras menu), 299
filters
Intel Dynamic filters, 313
Photoshop, 297, 298
Find File feature (Macintosh), 53
Find File option (Windows), 49
findCD( ) (example), 244
findEmpty( ), 131
FinderhiderXObject, 228
finding (see searching)
finishIdleLoad command, 280
firstIndent of member property, 383
fixedLineSpace of member property, 383
fixedRate of member property, 400
FixPaletteXObject, 575
fixStageSize property, 19, 162, 169
fixStageSize( ), 351
F-key shortcuts, 47
Flash Asset Xtra, 258, 261, 323, 401
Flash files
importing, 113, 115
sound, 494, 498
sprites (see direct-to-Stage sprites)
storage space, 258
vector graphics, 401
vector shapes, 402–406
Flash Player, Linux support, 200
flashRect property, 403
flatten sound, 485
FLC and FLI files, importing, 113
Flip Horizontal in Place (Transform menu), 59
flip tools (Paint window), 443
Flip Vertical in Place (Transform menu), 59
flipH of sprite property, 58, 70, 385
flipping (see animation)
flipV of sprite property, 58, 70, 385
float property, 559
floating-point coordinates, 151
floating-point numbers
performance, 290
platform differences, 213
floppy disk, loading onto, 294
FMA Online, xxii
focus (windows), 186, 189
Focus 3 Xtra, 573
Folder Icon Maker utility, 247
folders, Macintosh management, 53
folding sound, 485
font of member property, 381, 383, 388
Lingo access lacking, 142
FontList Xtra, 373
FONTMAP.TXT file, 19, 210, 373–376
fonts, 372–376
color chips for, 452
Director 7, 387
loading internal font map, 19
platform differences, 210
fontSize of member property, 381, 383
Lingo access lacking, 142
MIAWs, zooming, 185
fontSpacing of member property, 383
fontStyle of member property, 210, 381, 383, 388
   Lingo access lacking, 142
fontStyle property, 387
“Force Black and White Extension” plug-in, 419
foreColor of member property, 381, 397, 453–454
color depth and, 417
foreColor of sprite property, 15, 409, 453
color depth and, 417
foreground color, 11, 451
forget window command, 188
closeWindow event and, 180
disposing of MIAWs, 266
MIAWs not releasing memory, 192
restarting MIAWs, 173
format
digital video, non-QuickTime, 578–579
media (see media file formats)
menu items, 476
text, 372–376, 393
   chunk expressions, 385–387
forward looping (see looping)
4-bit (16-color) palettes, 205
frame keyword (go, play commands), 89
Frame Properties Palette dialog box, 430
frame property, 89, 174
frame rate
   absolute, 72
   animation, 62
   MIAWs, 173, 192
   setting, 71
   Tempo channel, 5, 71–72
frame scripts, 8
   Director 6/7 changes, 66
   Script channel, 8
frameLabel property, 75, 76, 95
framePalette property, 95, 96–98, 397, 432
   movies and, 171
frameRate of member property, 559, 560, 561, 566, 567
frameReady( ), 272
frames, 3–5
   accessing by name/number, 289
   checking sprite properties in, 133
   copying, inserting, deleting, 87–88
   cast members and, 128
   markers and, 96
   events, Director 6/7 changes, 66
   examining all, 99–100
   looping (see looping)
   markers for (see markers)
   moving playback head, 86
   navigating to (Lingo for), 88–92
   number readout when scrolling, 67
   offscreen buffers, 9, 265, 271
   pasting, 67
   playing speed (see frame rate)
   recording
      real-time, 67
      step-by-step, 67
   rendering to Stage, 9–11
   selecting entire, 67
   sprite spans, 7
   Director 6/7 changes, 63, 65
   transitions between (see transitions)
Frames Per Second option, 559
frameScript property, 95, 96–98
frameSound1, frameSound2 properties, 95, 96–98
movies and, 172
frameTempo property, 95, 96–98
movies and, 172
frameTransition property, 73, 95, 96–98
movies and, 172
framing digital video, 556
Framing option (Cast Member Properties), 379
Framing option (Field Properties), 379
Free Memory (Memory Inspector), 271
Free Rotate tool (Paint window), 444
freeBlock property, 254, 272, 273
preloading digital video, 541
   property differences and, 204
freeBytes property, 272, 273, 285
   property differences and, 204
freeing memory, 266
frontWindow property, 187
   Director vs. Projectors, 241
   movies and, 171
FTP, uploading Shocked files with, 343
Full Screen option, 166, 183
Projectors, 225, 228
fullColorPermit property, 206, 265, 294, 397, 428
function key shortcuts, 47

G
Gamma Fade Xtra, 455
gamma value, 205
garbage collection, 284
gauging performance, 288
gestalt selectors (Macintosh), 52
Get Info command, 247
Get Internet Explorer button, 336
Get QuickTime Pro movie, 549
Get Shockwave button, 337
getChannelCount( ), 529
getCommandLine( ), 232
GetCurrentFrame( ), 358
getError( ), 366, 520
generator
getErrorString( ), 520
getFinderInfo( ), 212
generateFrameRate( ) (example), 568
getFreeChannel( ), 529
getLatestNetID( ), 354
GetNetMessage( ), 362
getNetAddressCookie( ), 362
getNetErrorString( ), 362
GetNetMessage( ), 362
getNetOutgoingBytes( ), 362
getNetText( ), 354
generateNthFileNameInFolder( ), 212, 230, 246, 351
getNumberOfWaitingNetMessages( ), 363
generateOSdirectory( ), 212, 232
getPeerConnectionList( ), 363
getPixel( ), 410
getPlayStatus( ), 530
generatePrefs( ), 212, 346, 351
Director vs. Projectors, 241
generateSize( ) (example), 146
generateSndLength( ), 530
generateSndPosition( ), 530
generateStreamStatus command, 272
generateStreamStatus( ), 349, 355
generateVFWversion( ) (example), 543
Ghost ink effect, 446, 448
GIF Import Xtra, 322, 347
GIFs
importing, 113
storage space of, 257
(see also animated GIFs)
global variables
disposing of, 266
Stub Projectors, 221
globals property, 266
go command, 90–92
commands following, 90
history lists with, 91
go frame command, 95
go movie command, 169, 212, 352
GoToFrame( ), 358
GoToMovie( ), 358
gotoNetMovie( ), 342, 352, 355
gotoNetPage( ), 352, 355
Gradient ink effect, 446
Gradient settings dialog box, 436
gradientType property, 403
graphics
external castLibs of, 104
file formats, 207, 397–408
OS-level screen grabs, 54
performance issues, 292
width and height of sprites, 142
grid-based alignment, 164
GUI (see user interface)

H
halo effect, 450
halt command, 229, 241
Hand tool (Paint window), 438
handleCheckedItem( ) (example), 475
handler call stack, 283
“Handler not defined” error, 324
hardware
avoiding problems with, 23
CD-ROM drive, 244–246
compatibility problems, 202
internationalization, 215
monitor platform differences, 206
requirements for Director, 24–29
sound, 491, 530
header information (cast members), 19, 256
health awareness, 24
HEIGHT attribute, 341
height of member property, 158, 559, 562
height of rect property, 151, 161
height of sprite property, 15, 148, 158, 562
zero or negative value, 11
help
   Director 6 Help system, 33–34
   Director resources, xx–xxiii
   Windows OS references, 51
hidden cursor, 462
hiding/showing
cursors, 466
desktop, with Projectors, 228
Effects channels, 66
Lingo code, 230
MIAW titlebar, 178
MIAWs, 184
script preview area, 66
sprite path, 164
sprites, 164
Stage, 184
hierarchical menus, 478–482
highlight of member property, 461
Highlight When Clicked option, 458
highlighting text, 389–391
HighSpoolBufferMs command, 524
hilite command, 389, 390
hilite of member property, 142
hilite of sprite property (nonexistent), 142
history lists, 91
HLS color model, 408
holding area, MIAW as, 170
Home key, 47
hotSpot of member property, 470
hotspots, 407, 459
hourglass cursor, 466
HSB colormode, 408
HTML
   embedding Shockwave movies, 340
text as, 385
html of member property, 383, 385, 387
HTML Xtra, 385
hyperLink command, 392
hyperlinkClicked event, 392
hyperLinkRange command, 392
hyperLinks of member property, 383, 392
hyperLinkState command, 392
hyperLinkState
   formatting, 380
   Lingo for, 389–392
   Hypertext–General Behavior, 392
I
   I-beam cursor, 462
   icon resource files, 247
   Iconizer Xtra, 247, 302
icons for media types, 117
icons, customizing, 246–248
ID attribute, 341
idle events, 10, 14, 71, 280
displaying available memory, 285
paused movie and, 92
performance, 289
idle loading, 280–282
idleHandlerPeriod property, 280, 289, 293
idleLoadDone( ), 280
idleLoadMode property, 280–282
idleLoadPeriod property, 280
idleLoadTag property, 281
idleReadChunkSize property, 280, 281
if...then statements, xiv
ignoreWhiteSpace( ), 366
ilk( ), 152
IMA-compressed sound, 260
imageEnabled property, 403
images (see graphics)
implicitly loading cast members, 268
Import dialog box, 117, 120
“Import PICT File as PICT” option, 118, 399
importFileInto( ), 106, 123, 131, 310, 352
importing, 111–122
   animated GIFs, 400
   bitmaps, 398–400
   castLabs, 113
   custom palettes, 419–420
dynamically at runtime, 123
importing linked cast members, 146
   modes for, 117–119
tips for, 35
troubleshooting, 120–121
importLinks( ) (example), 146
“In a Window” option, 166, 183, 203
Projectors, 225
Include in Projector (Xtras dialog), 307, 316
Include Network Xtras option, 226, 305, 306, 322
“Include Original Data for Editing” option, 118, 400
indentation, inserting repeat loops and, 36
inequalities (for string comparisons), 214
INetURL Xtra, 322, 359
INF value, 213
inflate( ), 161, 185
Info button (Movie Xtras dialog), 307, 326
INI files, 50
initial load segment (movie), 19
ink effects, 445–450
- animation performance, 63
- performance issues, 292
ink of sprite property, 15, 381, 407, 447–450, 562
Insert menu, 42
insertFrame command, 95
inserting frames into Score, 87–88
inside( ), 156, 161
installing
- menus with installMenu, 476
- QuickTime, 549–550
- Shockwave (Win98), 335
Xtras after Director upgrade, 32
Xtras, detecting when installed, 326
installMenu command, 476, 477
Director vs. Projectors, 241
Instant Buttons and Controls, 461
insufficient memory, 121, 129, 282
external castLibs and, 105
Intel Dynamic filters, 313
interactivity
- Tempo frame rate and, 5
- user actions and sprite properties, 143–144
(see also runtime; user events)
interface (Director) (see user interface)
insertFrame( ), 565
internal cast members, 102
internal castLibs, 102
copying Behaviors from external castLibs, 103
listing, 152
management commands, 108
storage, 252
internal sounds, 490
compressed, 517
internationalization, 104, 215, 394
date and time presentation, 216
Internet
- Projectors for accessing content, 353–356
- streaming from (see streaming media)
- streaming performance, 290
Internet Explorer
- obtaining, 334, 336
- Shockwave support, 345–348
interrupting sound fades, 526
Intersect( ), 161
interval of member property, 470
Invert tool (Paint window), 444
invertMask of member property, 562, 566
isDirMMXloaded( ), 231
isMIAW( ) (example), 187
isPastCuePoint( ), 514, 520
isUsingQuicktimeStreaming( ), 565
isVRmovie of member property, 562
isVRmovie of sprite property, 562
J
jagged edges, 450
Java Export Xtra, 200, 313, 338
Java Player, 339
JPEG Import Xtra, 322, 347
JPEGs, storage space of, 257
jumping
- to frames or movies, 88–92
- to markers, 76
- to Score top, 67
K
kerning of member property, 383
kerningThreshold of member property, 383
Kersten, Kent, 302
Key Caps desk accessory, 376
keyboard events
delay command and, 92
Lingo interference with, 14
MIAWs and, 191
waiting for, 71, 93
(see also user events)
Keyboard Focus Sprite property, 380
keyboard shortcuts (see shortcuts)
keyboards, platform differences, 209
keyCode property, 210
keyDown event, 93
keyDownScript property
movies and, 172
keyframes, 4
Director 6/7 changes, 66
editing and manipulating, 81–85
keypad shortcuts (see shortcuts)

L
Label property (Cast), 111
label( ), 77
labelList property, converting to Lingo list, 76
labels, frame (see markers)
Lasso tool (Paint window), 438
last property, 386
lastChannel property, 292, 294
latency, 256, 497
layers, Photoshop documents, 398
leaky memory, 283
least common denominator
programming, 190
left of rect property, 151, 161
left of sprite property, 148, 152, 158, 562
leftIndent of member property, 383
length (see size)
length of Score vs. notation size, 4
length of string property, 381
“Let Windows manage my virtual memory settings” option, 255
libraries of cast members (see castLibs)
Library Palette window, 103
licensing
QuickTime, 549–550
Xtras, 305
Lighten ink effect, 446, 449
Lighten tool (Paint window), 444
Lightest ink effect, 446, 449
Limit Memory Size setting, 255, 271
line separators in menus, 475
Line tool (Paint window), 439
lineCount of member property, 381
lineDirection of member property, 408
lineHeight of member property, 381
lineSize of member property, 11
lineSize of sprite property, 158
lineSpace of member property, 383
Lingo
Cast and, 16
castLibs and cast members, 129–147
property-related syntax, 134–143
code security, external castLibs, 104
color-related, 207, 396–397
depth changes with, 413
palette properties, 432
cue points, 514
cursor-related, 468
dialog boxes, 483
digital video
memory management, 559
track control, 569–570
Director 7 improvements, xviii
events (see event handlers; events)
executing and updating Stage, 13–14
hiding code, 230
mailing list for (Lingo-L), xxiii
menu-related, 477
overriding Score, 14
performance tips, 288, 293
playback head and, 12
pop-up menus, 480
productivity tips, 36
Projectors vs. Director, 241
Score-related, 11–17, 88–98
channel control, 93
frame navigation, 88–92
playback head control, 92–93
recording Score, 94–98
waiting for events, 93
Shockwave and, 350–353
sound-related, 522–524
SWA-related, 520–522
volume control, 525
waiting for sound, 511
Lingo (continued)
  sprite property syntax, 134–143
  storage space, determining, 261
  text- and field-related, 380–394
  transition-related, 73
  translating to Java, 338
  Xtras vs., 300
  (see also puppeting)
Lingo (scripting) Xtras, 296, 298, 311, 323, 325, 326
Lingo Behavior Database, xxii
LINGO.INI file, 18, 200, 221
“Link to External File” option, 118, 323, 400, 490
linked media (see streaming media)
linked of member property, 400
linking
  cast members, 102
  at runtime, 123–124
  finding linked members, 145
  external castLibs, 105
  importing linked cast members, 146
  media types, 115–121
  advantages and disadvantages, 118
  Shockwave and linked media, 344
  sounds, 490
  compressing external sounds, 518
  troubleshooting, 120–121
Linux operating system, 200
listing
  castLibs, 132
  colors for cast members, 125
  commands contained in Xtras, 324
  installed Xtras, 326
  markers, 76
  operating performance, 289
Little Planet utilities, 302
live audio sources, 522
Load Once utility, 218
loaded of member property, 272, 562
LoadFont Xtra, 385
loading
  analyzing memory and, 269–276
  Cast, 265
  cast members, 9, 19, 123, 267–282
  performance and, 55
  determining what is loaded, 275
  digital video, 540
  displaying, 285
  idle loading, 280–282
  manual preloading, 276–280
  memory management and, 253
  Score, 4, 19, 265
  Shocked files onto web servers, 342–345
  Xtras, 18, 314–318
  (see also importing)
loc of sprite property, 7, 10, 15, 148, 149, 158
  (see also importing)
lm of sprite property, 58, 70, 143, 148
loop of member property, 61
  aligning cast members, 150
  locToCharPos( ), 389
  locV of point property, 15, 148, 158
  aligning cast members, 150
  locVToLinePos( ), 389
  locZ of sprite property, 58, 70, 143, 148
  Loop checkbox (Properties dialog), 61
  loop of member property, 61, 509, 560, 562
  loopBounds property, 562
looping, 59–61
  backwards, 4
  cast members and, 60
  digital video, 556
  film loops, storage space, 259
  in first or last frames, 4
  redrawing film loops, 61
  repeat loops
    event handling and, 14
    inserting without re-indenting, 36
    starting and stopping (simulating), 61
    static calculations in, 290
  loudness (volume, sound), 524–526
  lowercase characters, platform differences, 210, 214
  LowSpoolBufferMs command, 524

M
MACE compression, 487
machineType property, 200, 231, 235, 352
Macintosh, 195–216
Apple menu, 51, 473
customizing, 29
AppleScript language, 52
AutoPlay option, 248
backColor and foreColor settings, 454
caching Xtras, 318
CD-ROM, locating, 244–246
cross-platform strategy, 195–199
cursors, 462, 469
date and time, 215
determining platform, 234–238
developer shortcuts and tips, 51–54
development and delivery, 27
differences from Windows, 199–216
digital video
cross-platform issues, 548
detecting software for, 545
QuickTime installation, 549
QuickTime support, 541–543
hardware/software requirements, 26
icon customization, 246
importing media, 112–115
limitations to, 199
math operations, 213
media file formats, 207–209
memory and performance, 203, 253
menus, 210
monitors, 206–207
PICT files, importing, 121
porting checklist, 198
Projectors, 217, 222, 228
associating documents with, 250
Projectors and runtime issues, 202
rebooting options, 53
Shockwave, 321, 332
shortcuts, 36
shutDown, restart commands, 229
sound
compression, 518
mixing, 507
runtime sound card detection, 530
sound channels, 489, 496
Stage, changing, 182–183
starting desired Direction version, 32
streaming, performance of, 349
string comparisons, 214
system configuration, 29
system palette, 423
transitions, 207
window types, 178–179
Xtras and, 319
Xtras folder, 319, 324
too many files in, 291
Macromedia resources, xx
Macromedia SoundEdit, 297, 512, 535,
572
Macromedia Xtras, 301, 302, 308–314
MacroMix sound mixer, 500, 505–506
MACROMIX.DLL library, 18
MacroMix.X32 Xtra, 505
MacSoundManager mixer, 507
“Made with Macromedia” splash screen, 219
Magnifier tool (Paint window), 438
mailing lists on Director, xxii
Make Effects movie, 573
makeList(), 366
makeSubList(), 366
Mandell, Myron, 192
manually preloading, 276–280
manuals for Director, 33
map(), 161
mapMemberToStage(), 58, 70
mapping fonts, 373–376
mapping palettes, 423
mapStageToMember(), 58, 70
margin of member property, 381
Maricopa Director Web, xxii
markers, 17, 74–78
adding, 66
checking names, 76–78
copying/deleting frames and, 96
jumping to, 76
performance, 289
Markers window, 74
context-sensitive menu, 39
Marquee tool (Paint window), 438
Mask ink effect, 292, 448
mask of member property, 562, 566
masks for digital video, 566
Match First Movie option, 225
math operations, platform differences, 213
Math Xtras, 461
Matte ink effect, 292, 448
Mattes & Thumbs (Memory Inspector), 271
Maximum Visible property (Cast), 111
maxInteger property, 213
mci commands, 208, 530
mDispose command, 266
measuring performance, 288
media access, performance of, 290
media file formats
adding to movies, commands for, 122
for cast member properties, 138–141
importing, 112–115, 117–121
troubleshooting, 120–121
linking, advantages/disadvantages, 118
movies and castLibs, 109–110
platform dependencies, 207–209
Projectors, 226
sound file formats, 486
for sprite properties, 138–141
storage space and, 255–262
type of member property, 115–121
Media Interchange (see MIX Xtras)
Media Lab, 302
media of member property, 61, 95, 170
setting member properties at runtime, 138
media property, 562
media synchronization
lip sync, 72
Tempo channel for, 5
Media Type Icons property (Cast), 111
media windows, 45
MediaCleaner Pro, 539, 573
mediaReady of member property, 272,
352, 520
member( ), 387
member command, 131, 136
member of sprite property, 7, 11, 15,
134–136, 143
properties setting at runtime, 137
registration points, 150
stationary bitmap for, 61
member properties (see cast members,
properties)
memberNum of sprite property, 16,
134–136, 143
memberType of member property, 142
MemMon utility, 285
memory, 252–255
analyzing, 269–271
audio buffers, 255
budgetting, 263–267
checking if available, 272–276
data structure requirements, 265
digital video, 559, 576
digital video preloads, 540
freeing (disposing of objects), 266
importing during runtime, 123
insufficient, 121, 129
external castLibs and, 105
leaks, 283
MIAWs not releasing, 192
optimizing, 282–286
platform differences, 203
Projectors, 228
requirements, reducing, 286
Shockwave requirements, 332
troubleshooting errors, 282–286
virtual memory, 228, 253
Xtras cache file, 318
Memory Inspector, 269–271
context-sensitive menu, 38
Memory Limit (Memory Inspector), 271
memorySize property, 253, 254, 272
property differences and, 204
menu shortcuts (accelerators), 36–42
(see also shortcuts)
menuDisplayText of member property, 481
menus, 471–482
context-sensitive, 35, 37–39
platform differences, 199
Score window, 80
Stage and sprites, 81
cross-platform differences, 472
custom, defining, 471–475
disabled items, 476
formatting items in, 476
hierarchical and pop-up, 478–482
keyboard shortcuts, defining, 474
Lingo for, 477, 480
Message window
custom, defining, 38
D7.0 bug, 251
performance, 289
messages, 4
MIAW messages, 188–190
MHT Pop Up Xtra, 478
MHT-Icon Xtra, 247
MHTsearch Xtra, 303
MIAWs, 169–176
anomalies, 191–192
communications with Stage, 190
disposing of, 266
forgetting, stacking, 187
hiding/showing, 184
messages, 188–190
moving and cropping, 183
offscreen buffer size and, 265
opening, starting, closing, 31, 174, 187
panning and zooming, 185
platform differences, 204
properties of, 171
purging, 267
redrawing, 175
shared and unshared components, 171–174
source movies, 177
suggested uses, 170
for surrogate Stage, 170
tempo and transitions, 192
titlebar
hiding/showing, 178
setting text of, 177
as Tool Xtras, 170, 174
(see also movies; windows)
Microangelo Librarian utility, 247
Microsoft Internet Explorer
obtaining, 334, 336
Shockwave support, 345–348
MIDI sounds, 531
milliseconds property, 215
Mirror Horizontal (Transform menu), 59
Mirror Vertical (Transform menu), 59
missingFonts of member property, 384, 388
MIX Xtras, 297, 298, 308–310, 323, 398, 527
importing data types, 112
missing, 315
MIX Services Xtra, 316, 323, 347
MixBufferBytes command, 523
MixBufferCount command, 523
MixBufferMs command, 524
mixing sound channels, 496–507
Windows sound mixers, 500–507
MixIntPeriodMs command, 524
MixIntResolutionMs command, 524
MixMaxChannels command, 523
MixMaxFidelity command, 523
MixServiceMode command, 523
MixWaveDevice command, 523
mMessageList(), 565
MMX support, 237
(see also DirMMX Xtra)
MOD sound format, 531
modal of window property, 178
modified of member property, 272, 562
modifier keys, 209, 440
Modify menu, 42
monitor depth, 19, 225, 265, 408, 411–417
animation, 62
choosing, 411
converting to higher depths, 428
determining and changing, 233
palettes and, 5
performance and, 292
monitor-relative coordinates, 154–157, 162
monitors
deskTopRectList property, 232
gamma value, 205
platform differences, 206
using multiple, 24, 154, 206
mono sound, 485, 496, 519
mostRecentCuePoint(), 514, 520
Motion display mode, 67
mouse buttons, platform differences, 209
mouse coordinates, 159
mouse events
delay command and, 92
Lingo interference with, 14
MIAWs and, 191
modal of window property and, 178
waiting for, 71, 93
(see also user events)
mouse shortcuts, 35
mouseCast property, 389
mouseChar property, 386, 389
mouseDownScript property
movies and, 172
mouseEnter event, 389
mouseExit property, 159, 389, 468
mouseItem property, 386, 389
Index

mouseLeave event, 389
mouseLevel property, 562
mouseLine property, 386, 389
mouseLoc( ) (example), 160
mouseLoc property, 159, 468
mouseMember property, 389
mouseUpScript property, 172
mouseV property, 159, 390, 468
mouseWithin event, 390
mouseWord property, 386, 390
move( ), 131
move member command, 130
moveable property, 15
moveToBack command, 184, 187
moveToFront command, 184, 187
moveVertex( ), 406
moveVertexHandle( ), 406
moveWindow event, 189
movie property, 231, 241
movieAboutInfo property, 231
MovieCleaner (see MediaCleaner Pro)
movieCopyrightInfo property, 230, 231
movieFileFreeSize property, 132, 230, 252, 272
movieFileSize property, 272
movieName property, 172, 177, 231
Director vs. Projectors, 241
moviePath property, 172, 177, 231
movieTime of sprite property, 562, 568, 574
movieXtraList property, 322, 326
moving
cast members, 103, 127
constraining sprite movement, 164
digital video sprites, 540
external castLibs, 107
film loop center, 60
markers, 75
MIAWs, 183
playback head, 86, 88–92
sprites, pixel by pixel, 163
Stage aside (D6), 35
subtle repositioning during animation, 62
windows, 180
within Score, 85
(see also panning; positioning)
MPEG format, 578
MPEG Xtra, 539, 573
MPEG-3 compression algorithm, 261, 519
MPEG3 Xtras, 313
mPrint Xtra, 303
mRate of sprite property, 492, 562, 566
MUI Dialog Xtra, 313
dialog boxes, 483
MUI Xtra, 484
multi-channel sound, 496–507
Windows sound mixers, 500–507
multilanguage movies, 104
multimedia content on CDs, 531–533
Multimedia Tackle Box, 461
MultiMixer Xtra, 529, 573
multiple monitor support, 24, 206
colorDepth property, 233
deskTopRectList property, 232
platform differences, 206
multiSound property, 523
multiuser servers, 362
Multiuser Xtra, 361, 362
mute option, 525
muted sprite channels, 66

N
NAME attribute, 341
name of castLib property, 132, 172
name of member property, 388, 559, 562
name of menu property, 478
name of menuItems property, 478
name of window property, 177
name of Xtra property, 326
names
cast members, 7, 16, 105, 130–132, 289
caching, 130
searching for duplicates, 144
files (see filenames)
markers, 74, 76
checking, 76–78
menus, 473
movies, 177
Projectors, 203
resolving in Shockwave, 344
Stage, 177
windows, 175
NAN value, 213
narration, 216
native OS text selection, 390
navigable movies, 579
navigating (see moving)
Navigator (Netscape)
obtaining, 334, 336
Shockwave support, 345–348
negative coordinates, 148
netAbort(), 355
netDone(), 272, 352, 355
netError(), 355, 359–361
NetFile Xtra, 322, 326
netLastModDate(), 355
NetLingo Xtra, 322, 326, 353
NetManage WinSock Lib Xtra, 322
netMime(), 355
netPresent(), 326, 352, 355
netPresent property, 352
Netscape Navigator
obtaining, 334, 336
Shockwave support, 345–348
Netscape Now button, 336
netStatus(), 352, 355
Director vs. Projectors, 241
netTextResult(), 355
netThrottleTicks property, 204, 294, 349
network errors, 359–361
network-related Xtras, 308, 322
castLibs, 105
number of castLib property, 132
number of castLibs property, 132, 172
number of member property, 135
Shared Cast and, 106
number of members of castLib property, 132
number of menuItems of menu property, 478
number of menuItems property, 477
number of menus property, 477, 478
number of the member of sprite property, 135
number of Xtra property, 326
number property, 386
numbers
cast members, 6, 289
calculation performance, 290
numbers (continued)
  frames, readout during scrolling, 67
 random, 214
 sound channels, 496
 numChannels of member property, 519, 520
 numeric keypad shortcuts (see shortcuts)

O
  obeyScoreRotation of member, 70
  obeyScoreRotation of member property, 58
  object disposal, 266
  object movies, 579
  <OBJECT> tags, 340–342
  obtaining
    Shockwave, 334
    Shockwave test software, 334
    Shockwave, user-side, 335, 337
    web browsers, 334
    Xtras, 301–305
  office environment, 24
  offscreen buffer, 9, 265, 271
  offset( ), 161
  “Ok for all movies” option, 108

OLE
  files, importing, 114
  platform dependencies, 209
  on activateWindow handler, 189
  on beginSprite handler, 9, 133
  on closeWindow handler, 180, 189
  on cuePassed handler, 515, 516
  on deactivateWindow handler, 189, 241
  on endSprite handler, 465
  on enterFrame handler, 6, 94
  on EvalScript handler, 358, 359
  on exitFrame handler, 92
  on hyperlinkClicked handler, 392
  on idle handler, 14, 71, 280
    displaying available memory, 285
    performance, 289
  on in Lingo code, xiii
  on keyDown handler, 93
  on menuitemSelected handler, 481
  on mouseEnter handler, 389
  on mouseLeave handler, 389
  on mouseWithin handler, 390
  on moveWindow handler, 189
  on new handler, 133
  on openWindow handler, 189
  on prepareFrame handler, 9, 19
    transitions and, 6
  on prepareMovie handler, 19, 189, 201
    changing sprite properties in, 143
  on resizeWindow handler, 189
  on startMovie handler, 189
    changing sprite properties in, 143
  on startUp handler, 189
    Stub Projectors, 221
  on stopMovie handler, 189
  on StreamStatus handler, 355
  on updateStage handler, 9
  on zoomWindow handler, 189
  one-frame movie, 11
  Onion Skin window
    context-sensitive menu, 38
    Onion Skinning, 150
  online help (Director 6), 33–34
  online resources on Director, xx
  onStage( ), 157
  open command, 169, 188, 212
  Open Markers window, 75
  open window command, 169, 174, 177, 189
  open...with command, 213
  openDA command, 212
  OpenDBC Xtra, 303
  openFile command, 213
  opening Stage window (D7), 42
  opening windows, 42–44, 174, 187
  openResFile command, 212, 468
  openWindow event, 189
  openXlib( ), 213, 314
  operating systems, 195–216
    browser scripting support, 357
    context-sensitive menus, 199
    cross-platform strategy, 195–199
    cursors, 211
    date and time presentation, 215
    determining platform, 234–238
    differences between, 199–216
    digital video issues, 548
    filenames, 211–213
    font mapping across platforms, 373
    graphics file formats, 207
    menu differences, 472
Index 609

Palettes, 417–430
  animation limitations, 62
  color chips, 451–454
  Color Palettes window, 433
  conflicts, 424
  custom, 121, 419–420
  digital video, 575
  importing, 113, 419–420
  interface options, 429
  Lingo properties for, 432
  MIAWs and, 192
  operations on, 423
  Paint window (see Paint window)
  platform differences, 204
  reserving colors in, 425
  storage space for, 259
  troubleshooting, 425–430
  types of, 420–423
  Xtras for, 454
  panMIAW( ), 185, 186
  panning MIAWs on Stage, 185
  panoramas, 579

O

Palette channel, 5
  MIAWs and, 192
  properties, 430–433
  palette effects, 70
  fades, 205, 431
  transitions, 430
  palette flashes, 414, 430
  palette of member property, 397, 423, 432
  paletteIndex( ), 410
  paletteMapping property, 397
  paletteRef of member property, 397, 423, 432
  palettes, 417–430
  animation limitations, 62
  color chips, 451–454
  Color Palettes window, 433
  conflicts, 424
  custom, 121, 419–420
  digital video, 575
  importing, 113, 419–420
  interface options, 429
  Lingo properties for, 432
  MIAWs and, 192
  operations on, 423
  Paint window (see Paint window)
  platform differences, 204
  reserving colors in, 425
  storage space for, 259
  troubleshooting, 425–430
  types of, 420–423
  Xtras for, 454
  panMIAW( ), 185, 186
  panning MIAWs on Stage, 185
  panoramas, 579

Page Up, Page Down keys, 47
  pageHeight of member property, 381
  Paint brush settings dialog box, 435
  Paint bucket tool (Paint window), 439
  Paint window, 435–451
    bitmaps, wrong sizes for, 121
    button shortcuts, 45
    centering sprites in, 164
    color chips for, 451
    context-sensitive menu, 39
    effects tools, 443
    inks (see ink effects)
    preferences, 435
    registration point and, 148, 150
    shortcuts, 442–443
    text tool formatting, 372
    tips and tricks, 440–442
    tools of, 436–440
    troubleshooting, 450
  Paintbrush tool (Paint window), 439
  PALETTE attribute, 341
  PaintColor channel, 5
  MIAWs and, 192
  properties, 430–433
  palette effects, 70
  fades, 205, 431
  transitions, 430
  palette flashes, 414, 430
  palette of member property, 397, 423, 432
  paletteIndex( ), 410
  paletteMapping property, 397
  paletteRef of member property, 397, 423, 432
  palettes, 417–430
  animation limitations, 62
  color chips, 451–454
  Color Palettes window, 433
  conflicts, 424
  custom, 121, 419–420
  digital video, 575
  importing, 113, 419–420
  interface options, 429
  Lingo properties for, 432
  MIAWs and, 192
  operations on, 423
  Paint window (see Paint window)
  platform differences, 204
  reserving colors in, 425
  storage space for, 259
  troubleshooting, 425–430
  types of, 420–423
  Xtras for, 454
  panMIAW( ), 185, 186
  panning MIAWs on Stage, 185
  panoramas, 579
paragraph of member property, 384
`parseString( )`, 366
`parseURL( )`, 366
parsing text, 289
partial character sets, 388
Partition Size (Memory Inspector), 271
pass command, 171
`pasteClipBoardInto( )`, 352
pasting
  - frames, 67
  - sprites, 67
  (see also copying)
path resolution in Shockwave, 344
pathName property, 213, 231
  - movies and, 172
pattern chips, 451
pattern color chip (Paint window), 452
pattern of member property, 397
Pattern settings dialog box, 436
Pause When Window Inactive option, 173
`pause( )`, 401, 521
SWAs and, 521
`pausedAtStart of member property`, 541, 560
`pauseRawSound( )`, 550
pausing movies
  - for cue points, 72
  - for number of seconds, 71
  - for specified number of ticks, 92
  - for user events, 71, 93
pausing sound fades, 526
pausing SWAs, 521
Peak LE utility, 297, 485, 536
Pencil tool (Paint window), 439
Pentium Processor, checking for (example), 257
Penworks, 302
`percentPlayed of member property`, 521
`percentStreamed of member property`, 521
performance, 20, 287–295
  - animation optimization, 61–63
  - avoiding hardware/software problems, 23
  - bitmaps, 399
  - D7 dot syntax, 290
digital video, 539–541, 577
digital video compression, 575
direct-to-Stage sprites, rendering, 10
external castLibs and, 103
idle loading, 280–282
Java Export Xtra, 339
Lingo issues, 288, 293
loading cast members, 55, 267
math operations, 213
measuring, 288
media and disk access, 290
memory analysis, 269–276
memory optimization, 282–286
menu installation, 476
network operations, 353
Paint Window and, 450
platform differences, 203
playback head direction and, 4
Projectors, 219
runtime environment, analyzing, 230–238
sampling rates, 208, 485
streaming media, 349
string manipulation, 258
SWA decompression, 519
testing, 27
throughput, 255, 262
timer resolution, 215
tips (see tips and tricks)
transition execution, 72
transitions, 551
virtual memory, 253
Perspective tool (Paint window), 444
Photocaster Lite Xtra, 314
Photocaster Xtra, 150, 302, 455
Photoshop (Adobe)
  - custom palette creation, 419
  - filter Xtras, 297, 298
  - importing CLUT files, 114
  - paint layers, 398
  - registration points, 150
physical health concerns, 24
Physical Memory (Memory Inspector), 271
PICS files, importing, 113
PICT files
  - importing, 113, 114, 121, 399
  - platform dependencies, 209
  - storage space, 257
PICT Import Export Xtra, 121
picture of member property, 352, 381, 384, 394, 445
importing and, 123
setting member properties at runtime, 138
updating loops when changing, 61
picture of the stage property, 54, 169, 176, 417
picture of window property, 54
Picture Xtra, 455
pointToParagraph( ), 386
Pleaenitz, Daniel, 250
Planet Color Xtra, 455
platform (see operating systems)
platform property, 200, 231, 234, 352
Director vs. Projectors, 241
play command, 88
subsequent commands, 91
play done command, 88, 91, 267
memory leaks from not using, 284
sprite properties and, 172
“Play Every Frame (No Sound)” option, 558, 567
Play Every Movie option, 225
play movie command, 213
not using play done with, 284
sprite properties and, 172
tell command and, 91
“Play While Downloading Movie option”, 348
play( ), 358, 521
playback environment (see Projectors)
playback head, 3, 78–88
controlling (Lingo for), 92–93
direction of, performance and, 4
indicator, Director 6/7 changes, 65
Lingo and, 12
MIAWs, 173
moving, 86, 88–92
playback modes for animated GIFs, 401
playback, digital video (see digital video)
playback, sound (see sound)
playBackMode of member property, 401
Player (D7.0), 226–227
playing sound (see sound)
playing speed (see frame rate)
playRaw( ), 530
playRegular( ), 530
playVoiceOver( ) (example), 493
Plug-ins folder (browsers), 345
PLUGINSAGE attribute, 341
point( ), 161
pointInHyperlink( ), 392
points, 151–154
distance between, calculating, 154
manipulating, 160–162
pointToChar( ), 384, 386
pointToItem( ), 384, 386
pointToLine( ), 384, 386
pointToParagraph( ), 384, 386
pointToWord( ), 384, 386
Polygon tool (Paint window), 440
PopupMenu Lite Xtra, 314, 478
PopupMenu Xtra, 314
pop-up menus, 478–482
Popup Xtra, 314, 478–482
porting between platforms (checklist), 198
position
coordinates, 151–163
properties of, 158–160
Stage- vs. monitor-relative coordinates, 154–157, 162
cursor, 470
digital video, 540
monitors, determining, 232
sound, 507
on Stage, 148
aligning cast members, 150
Stage, 163–165, 167, 183, 225
system cursor, 470
vector shape vertices, 404–406
windows, 183–186
positioning (see moving)
postNetText( ), 356
Power Macintosh Only option, 222
PowerPoint, 76, 115
PowerPoint Import Xtra, 313
Precision Xtra, 454
preferences
castLibs, 111
Director, 68
Paint window, 435
Shockwave, 345–348
Pre-Fetch option, 348
preLoad command, 277
preLoad of member property, 277–279, 540, 541, 560
preLoadBuffer(), 277, 521
preLoadCast command, 277
preLoadEventAbort property, 277
preloading digital video, 540
preloading manually, 276–280
preLoadMember command, 277–279
preLoadMode of castLib property, 132, 253, 278
preLoadMovie property, 175, 278
preloadNetThing(), 278, 356, 359
preLoadRAM property, 278, 279, 280, 540, 576
preLoadTime of member property, 277, 278, 521
prepareFrame event, 9, 19
delay command and, 92
frame scripts and, 8
transitions and, 6
prepareMovie event, 19, 189
changing sprite properties, 143
preRender of member property, 384
preRender property, 293
pre-rolling (preloading DV), 540
“Preview in Browser” option, 332
Preview in Browser (File menu), 330, 332
price of Xtras, 304
printFrom command, 352
printing
marker comments, 76
performance, 289
Xtras for, 303
PrintOMatic Lite Xtra, 314
PrintOMatic Xtra, 303
privacy, 230
production management, 21–48
development environment, 25–26
shortcuts (see shortcuts)
system configuration, 29–33, 198
testing environment, 26–29
productName property, 231
productVersion property, 228, 231, 236, 352
Projec32.dll file, 228
Projec32.skl file, 228
Projector Options dialog box, 223
Projector.INI file, 200
Projectors, 217–230, 329
accessing Internet content, 353–356
creating, 221–228
customizing icons, 246–248
differences from Director, 238–243
loading onto floppy disk, 294
locating user’s CD-ROM, 244–246
memory requirements, 254, 263
palettes, 422
platform differences, 202
starting, 218
automatic startup, 248–249
with document files, 250
Stub Projectors, 219–221
utilities, 243–251
preventing user from aborting, 243
Xtras with, 224, 226, 322–327
bundling Xtras into Projectors, 203, 226, 324–326
Director vs. Projectors, 240
(see also Shockwave)
Projector.rsr file, 228
Projector.skl file, 228
properties
  cast member properties, 133–143
  changing, 143
  complete list of, 138–141
  setting, 137
  disposing of property variables, 266
  hierarchy of, 19
  idiosyncracies, 142
  listed by object, 20
  movies (MIAWs), shared and unshared, 171
  runtime environment properties, 230–238
  setting, performance of, 289
  size-related, 261
  sprite properties, 7, 133–144
  changing, 4, 64, 143
  complete list of, 138–141
  Director 6/7 changes, 65
  movies and, 172
  setting, 137
  transformation-related, 70
  user events and, 143–144
  Stage-related, 168
  window properties, 176–187
“Property not found” error, 574
Index

property of sprite property, 94
protected format (movies, castLibs), 109
protecting external assets, 230
proxyServer( ), 356
pull-down menus (see menus)
puppet of sprite property, 14, 64
movies and, 172
puppet sprites, 7, 267
puppeting, 7
automatic vs. manual, 14
Effects and sprite channels, 94
empty sprite channel, properties, 143
troubleshooting, 15
(see also Lingo)
puppetPalette command, 94, 397, 432–433
MIAWs and, 192
movies and, 171
palettes and, 421
puppetSound command, 94, 492
puppetSprite command, 14–15, 64, 94
puppetTempo command, 71, 94
movies and, 172
puppetTransition command, 74, 94, 172
movies and, 172
Purge button (Memory Inspector), 269, 285
purgePriority of member property, 253, 267, 268, 272, 541, 560, 562
checking, 285
purging items, 267
push buttons, 115, 459–461
put command, 242, 285
performance of, 288

Q
QD3D (QuickDraw 3D), 581
QMIX sound mixer, 501
QT Gallery 1.0, 573
QT Xtra Properties dialog box, 555
QT3 encryption keys, 230
QT3 Extension (Apple), 541
QT3 Pro, 573
QT3 Xtra, 546
QT3Asset.X32 Xtra, 505
QT3Mix sound mixer, 501, 505, 505–506
qtRegisterAccessKey( ), 565
qtUnRegisterAccessKey( ), 565
QTVR 1.0 Xtra, 152
QTVR 2 Sprite Xtra, 580
QTVR Authoring Studio, 580
QTVR files, importing, 115
QTVR format, 579
QTVR Xtra, 580
QTVREntry( ), 581
QTVRExit( ), 581
QTVROpen( ), 581
quad of sprite property, 58, 70, 293, 294, 385
QuickDraw 3D Xtra, 581, 582
QuickDraw shapes (see shapes)
QuickKeys utility, 53
QuickTime
audio tracks (see sound)
cue point support, 512
importing files, 115
installation and licensing, 549–550
platform dependencies, 208
QT3Mix sound mixer, 501, 505
resources for, 572
versions of, 541–543
QT3 Pro vs. QT3, 549
video formats other than, 578–579
Xtras needed for, 323
QuickTime 3 Asset Xtra, 264
QuickTime 3 Pro, 573
QuickTime 3 Xtra, 547
QuickTime Asset Xtra, 541, 546
QuickTime PowerPlug, 545
QuickTime window
content-sensitive menu, 38
QuickTime Xtra Properties dialog box, 554
quickTimePresent property, 544, 545
quickTimeVersion( ), 544–546, 565
quit command, 229
quitting, preventing user from, 243

R
radio buttons, 115, 459–461
RAM (see storage space)
ramNeeded( ), 259, 272, 273
random( ), 214
random numbers, 214
rate (see speed)
RealAudio Xtra, 522
realtime playback (see streaming media)
realtime recording, 67
RealVR Xtras, 580
RearWindow XObject, 228
rebooting options for Macintosh, 53
recordFont(), 388
recording
  real-time, 67
  Score, Lingo for, 94–98
  step-by-step, 67
recovering corrupted files, 129
Rect(), 162
rect of member property, 158, 381, 559, 563
rect of sprite property, 15, 148, 159, 379, 382, 408, 563
  scale and, 404
rect of the stage property, 162, 184
  movies and, 172
rect of window property, 162, 183–186
  redrawing MIAWs, 175
Rectangle tool (Paint window), 439
rects, 151–154
  calculating area (example), 153
  manipulating, 160–162
recursion, 283
RedBook Audio, 532
RedBook format, 531–533
redrawing
digital video, 578
  film loops, 61
MIAWs, 175
  (see also alignment, position on Stage)
ref keyword, 384, 387
references for further reading (see help)
references, Score (see sprites)
registered trademark symbol, 376
registering Xtras, 314–318
Registration point tool (Paint window), 438, 440
registration points, 121, 148–150, 407, 440
  aligning cast members, 150
  cursors, 464
Registry file (Windows), 50
regPoint of member property, 60, 148, 150, 152, 158, 468
  updating loops when changing, 61
regPoint of sprite property, 403, 440
  checking if on Stage, 156
regPoint property, 563
relinking external castLibs, 105
Remap Palettes When Needed option, 423, 428
remapping palettes, 423
removable media, 28
renaming (see names)
rendering frames to Stage, 9–11
repeat loops
event handling and, 14
  inserting without re-indenting, 36
“The requested object could not be found on the server” error, 359
resampling, 485
reserved colors, 62, 204
reserving colors in palettes, 425
Reset All (Transform menu), 59
“Reset Monitor to Match Movie’s Color Depth” option, 203, 225
Reset Rotation and Skew (Transform menu), 59
Reset Width and Height (Transform menu), 59
resetAllCursors() (example), 467
resizeWindow event, 189
resizing (see size)
resolution
  monitor, 206
  NTSC output, 553
timer, 215
resource forks, 52
resources on Director, xx–xxiii
resources on Windows OS, 51
restart command, platform and, 229
result property, 278–279
resume(), 401
Reveal ink effect, 446
Reverse ink effect, 446, 447
rewind(), 358, 401
rewinding digital video, 571
RGB color model, 408–410
rich text, 369–376
  advantages and disadvantages, 370
  appearance and attributes, 370–376
attributes for, 379–380
in cursors, 469
fonts and formatting, 372–376
importing files, 113
Lingo for, 380–389
properties without Lingo access, 142
storage space, 259
troubleshooting, 393
width/height of sprite properties, 142

(see also text)
Rich Text Cast Member Properties
dialog box, 379
right mouse button (Windows), 49
right of rect property, 151, 162
right of sprite property, 148, 159
rightIndent of member property, 384
RLE compression, 257, 398
RobinHood Xtra, 573
rollover property, 143
romanLingo property, 215, 289, 394
Rotate Left (Transform menu), 59
Rotate Right (Transform menu), 59
Rotate tools (Paint window), 444
rotation, 57, 451
(see also animation)
rotation of member property, 58, 70, 563
rotation of sprite property, 58, 70, 385, 403, 451, 563
Round Window WDEF, 178, 179
Row Width property, (Cast), 111
RSX service, 501
rsxDontUseDirectSound command, 505, 523
rtf of member property, 384, 385, 387
RTF versions of text, 385
Run options (Windows), 49
runMode property, 200, 231, 235, 352
Director vs. Projectors, 242
runtime (dynamic activity)
attaching castLibs, 106
cast member creation, 133, 284
checking for DV software, 543–546
drawing, 407
importing, 125
linking to cast members, 123–124
modifying cast members, 106
setting member properties, 137
sound card detection, 530
runtime environment
analyzing, 230–238
Director vs. Projectors, 238–243
error checking, 233
Projectors (see Projectors)
runtime issues, cross-platform, 202
RXSMix sound mixer, 501
S
Safe Mode (Windows 95), 50
safePlayer property, 231, 352
sampleRate of member property, 509, 521
sampleSize of member property, 509
sampling rates, 208, 485
Sandau, Mark, 420
Save and Compact (File menu), 252
“Save as Java” feature, 338
save castLib command, 106, 132
Director vs. Projectors, 242
saveBitmap of member property, 384
saveBitmap property, 293
saveMovie command, 132
Director vs. Projectors, 242
platform differences, 203
saving movies and castLibs, 110
scale (see size)
scale of member property, 58, 70, 403, 556, 563
scale of sprite property, 58, 70, 403, 404, 563
scaleMode property, 403, 404
scaling modes, 404
Score, 3–5, 63–69
animation with, 57–59
corrupted, 291
Effects channels (see Effects channels)
external castLibs references, 107
frame manipulation, 87–88
incorrect references to, 129
jumping to top, 67
keyframes, 4, 66
editing sprite paths, 81–85
length of, 4
Lingo for, 11–17, 88–98
loading, 4, 19, 265
marking frames (see markers)
Score (continued)
  navigating, 85
  notation size, 4
  overriding with Lingo, 14
  performance issues, 291
  playback head (see playback head)
  productivity tips (see tips and tricks)
  purging data from, 267
  recording, 94–98
  references (see sprites)
  sound in, 491–494
    (see also sound)
  sprite channels (see sprite channels)
    troubleshooting, 98–100
  views of (see Score window)
Score channels
  colorization, 452
  number of, performance and, 29
Score Palette channel, 421
score property, 95, 170
score scripts, 8
  (see also behaviors)
Score window
  changing view, 85
  context-sensitive menu, 39, 80
  customizing appearance, 68
  Director version differences, 65
  opening multiple, 79
scoreColor of sprite property, 95, 397
scoreSelection property, 95, 97–98, 397
  Director vs. Projectors, 242
scrap tags, 87, 128
Scrapbook, importing, 113
Screen Buffer (Memory Inspector), 271
screen grabs, 54
screen savers, Windows, 232, 249
screenshots in this book, xv
Script channel, 8
Script display mode, 67
script of menuitem property, 478
script preview area, hiding/showing, 66
Script window
  button shortcuts, 45
  context-sensitive menu, 38
  opening multiple, 44
scripting Xtras (see Lingo Xtras)
scriptNum of sprite property, 95
ScriptOMatic Lite Xtra, 314
scripts
  changing in Effects channels, 69
  colorizing, 36
    color chips, 452
  external castLibs of, 104
  for cast members (see cast scripts)
  for frames (see frames scripts)
  for scores/sprites (see behaviors)
  storage space for, 259
scriptText of member property, 242
scriptType of member property, 133
ScrnUtil Xtra, 455
ScrnXtra, 302
scrolling
  frame number readout, 67
  menus, 210
  text, 379
scrollTop of member property, 382
SE16 format, 487
searchCurrentFolder( ), 352
searching
  for cast members, 124, 130
  for duplicate member names, 144
  for installed Xtras, 326
  for linked cast members, 145
  for text, performance and, 289
  for user's CD-ROM, 244–246
  for Xtras, 301
searchPath property, 213, 290, 352
searchPaths property, 213, 290, 352
security, external castLibs and, 104
Select Used Colors option, 431
selectedSpec of member property, 481
selectedText of member property, 384,
  387, 481
selection of castLib property, 132, 242
selection of member property, 387, 390,
  395
selection property, 390
Selection tool (Paint window), 438
selEnd property, 390
selStart property, 390
sendAllSprites( ), 289
sendNetMessage( ), 363, 364
serialNumber property, 231, 242
SerialPort XObject, 314
servers (see web servers)
setFilterMask command, 213
Index 617

setFinderInfo( ), 212, 251
setHyperLink( ), 392
setMonitorDepth( ) (example), 414
SetMouse Xtra, 470
SetNetBufferLimits( ), 363
setNetMessageHandler( ), 363
setPixel( ), 410
setPref( ), 213, 242, 346, 353
setSoundTime( ), 530
setTrackEnabled( ), 555, 565, 569
Shape Xtra, vector shapes and, 402
shapes, 398, 406–407
as buttons, 459
performance, 293
registration points, 149, 150
storage space, 258
vector shapes, 402–406
shapeType of member property, 133,
407
Shared Cast, 106–107
Sharkbyte Killer Transitions Xtra, 73
Shift key (see shortcuts)
Shockwave, 328–366
browser support for, 345–348
communicating with browsers, 356–359
creating movies, 332
cut and paste, 394
D7 features, 361–366
development requirements, 26
Director vs., 349–353
importing Audio files, 114
importing Flash files, 113, 115
importing media, 112
installing on Win98, 335
Lingo for, 350–353
mailing list for (ShockeR), xxiii
MIAWs not supported, 191
MIX Xtras and, 310
movie size, 167
network errors, 359–361
obtaining, 334, 337
performance, 290
platform-dependency, 331
preloading, 277
references for, 329
running movies on web pages, 339–342
Smart Shockwave utility, 337
sound, Xtras needed for, 528
(see also SWA)
storage space and compression, 261
streaming playback, 330, 348–349
SWA Xtras, 297, 298, 310, 313,
321–322
SWA Compression Xtra, 321
testing, software for, 334
uploading files to web server, 342–345
version, checking if adequate, 338
versions 7 improvements, xix
Shockwave ActiveX control, 330
“Shockwave for Audio Settings” option,
299
Shockwave for Flash player, 401
Shockwave option (Project creation),
227
shopping for Xtras, 303–305
shortcuts, 35, 36–48
Cast window, 125–127
defining for menu items, 474
kepad and special keyboard keys, 46–48
Macintosh-related, 54
menu accelerators, 36–42
opening/closing windows, 42–44
Paint window, 442–443
platform differences, 199, 210
quitting, preventing user from, 243
Score, 79
frame manipulation, 87
moving around in Score, 85
moving playback head, 86
sprite channel manipulation, 83
Stage, 83
vector shape vertex manipulation,
404
Windows operating systems, 51
(see also context-sensitive menus)
Show Cast Member Script Icons
property (Cast), 111
Show Controller option, 558
Show Placeholders option, 348
Show Title Bar option, 167, 183
showCastLibTypes( ) (example), 132
showing (see hiding/showing)
showLoaded( ) (example), 275
showProps( ), 402
showResFile(), 212, 468
showXlib(), 213, 326
shuffling sprite channels, 66
shutDown command, platform dependence, 229
16-bit palettes, 411
16-color palettes, 205
size
animated objects, performance, 62
bitmaps, 257, 398, 442, 450
resized bitmap sprites, 63
wrong size in Paint window, 121
compressed (see compression)
cursors, 469
digital video, 538, 555
digital video scale, 556
file size, determining, 146
marker names and comments, 74
monitors, 232
offscreen buffer, 265, 271
properties related to, 261
scaling modes, 404
Score notation
length vs., 4
loading entire Score and, 4
sprite channels and, 5
Shockwave movie, 167
sounds, 485, 509
sprites, 163–165
Stage, 19, 63, 163–165, 167, 183, 225
subtle changes during animation, 62
text, zooming MIAWs and, 185
vector shapes, 404
window, 183–186
(size also storage space)
size of member property, 256, 258, 261, 273, 495
Director vs. Projectors, 242
externally linked files, 146
(size also storage space)
size property, 563
skew, 57
skew of sprite property, 58, 70, 385
Skew tool (Paint window), 444
Slim Projectors, 227, 294
reading Xtras, 320
Smart Shockwave utility, 337
Smear ink effect, 446
Smooth ink effect, 446
Smooth tool (Paint window), 444
Smudge ink effect, 446
software
avoiding problems with, 23
internationalization and, 215
requirements for Director, 24–29
Sorenson Video Codec, 575
sorting
cast members, 124
performance and, 289
sound, 485–536
bit depth, 485
buffers for, 255
channels (see Sound channel; sound channels)
compressing (see SWA)
digital video, 551, 576
ing, utilities for, 535
Enhanced CDs (ECDs), 531–533
external castLibs of, 104
file formats, 208, 486
importing, 113, 114
interface options, 508–510
Lingo for, 522–524
SWA, 520–522
volume-related, 525
lip sync, 72
live audio, 522
memory leaks from not unloading, 284
MIAW conflicts with Stage, 191
MIDI, 531
operations, comparisons of, 494–496
output devices, 497
performance, 290, 291
playback methods, 488–489, 491–496
position and playback control, 507
runtime sound card detection, 530
sampling rate, 208, 485
selecting from pop-up menu
(example), 481
Shockwave (see SWA)
size of, 485, 509
storage space, 260
streaming (see streaming media)
testing, 70
timing (see synchronizing media)
troubleshooting, 533–535
user interface issues, 488

618 Index
voice narration, 216
volume levels and fading, 524–526
when triggered, 10
Xtras for, 310, 526–530
Sound Cast Member Properties dialog box, 509
Sound channel, 6, 486
conflicts, 491
sound channels, 485, 486
digital video and, 551
mixing, 496–507
  Windows sound mixers, 500–507
number conflicts, 496
sound close command, 522
sound devices/drivers, 491
Sound dialog box, 70
sound fadeIn command, 525, 526
sound fadeOut command, 525, 526
Sound Forge XP, 513, 536
Sound Import Export Xtra, 310, 316, 323
Sound Mixer Xtras, 297
sound of member property, 525, 559, 563, 570
sound playFile command, 102, 213, 490, 492
sound stop command, 523
soundBusy( ), 509, 522
soundChannel of member property, 496, 521
soundDevice property, 505–507
soundDeviceList property, 505–507
SoundEdit 16 (Macromedia), 297, 512, 535, 572
soundEnabled property, 523, 525
soundKeepDevice property, 498
soundLevel property, 208, 523, 525, 526
SoundLevel0...SoundLevel7 settings, 524, 526
sourceRect of the stage property, 163
sourceRect of window property, 163, 183–186
spans, sprite, 7
  Director 6/7 changes, 63, 65
  special characters, 375–376
  Speech Recognition, 529
  speed
digital video, 539, 558, 567
frame play (see frame rate)
sound sampling rate, 208, 485
streaming media data rate, 256, 486
preloading and, 279
transition execution time, 72
spell-checker Xtra (rumored), 314
SpoolBufferAlloc command, 524
SpoolBufferCount command, 524
Spread ink effect, 446
Sprite Asset Xtras, storage space, 261
sprite channels
  context-sensitive menus, 80
  controlling with Lingo, 93
  empty, puppeting and setting properties, 143
  examining all, 99–100
  Lingo with, 17
  manipulating, 83
  number of, 65, 170
  ordering, 163
  Score notation size and, 5
  in scoreSelection list, 97
  selecting entire, 67
  shuffling forward/backward, 66
Sprite Inspector
color chip for, 452
context-sensitive menu, 38
sprite paths
  Director 6/7 changes, 66
  editing and manipulating, 81–85
  hiding/showing, 164
sprite properties, 7, 133–144
  changing, 4, 64, 143
  complete list of, 138–141
  Director 6/7 changes, 65
  idiosyncracies, 142
  movies and, 172
  setting, 137
  setting fileName of member property, 124
  transformation-related, 70
  user events and, 143–144
Sprite scripts, 8
  Director 6/7 changes, 66
  (see also behaviors)
Sprite Toolbar, 66
color chip for, 452
context-sensitive menu, 39
Sprite Xtras, 297, 298, 311, 323
  loading, 19
memory leak with external castLibs, 284
missing, handling, 315
SpriteBox command, 159
spriteNum of me property, 137, 143
sprites, 6–7
  aligning and sizing, 163–165
  animating (see animation)
  auto-puppeting, 14, 137
  bounding box, checking if on Stage, 157
cast member numbers, 6, 289
changing cast members and, 8
channels for (see sprite channels)
context-sensitive menus, 81
coordinates for, 152, 158
copying, 87, 128
cursors for, 464
digital video, 538, 578
direct-to-Stage, when rendered, 10
dragging (example), 160
erasing automatically, 10
events, Director 6/7 changes, 66
hiding/showing, 164
inks for, 447–450
not appearing on Stage, 10
overlay, color chips for, 452
pasting, 67
properties (see sprite properties)
puppet sprite, 7
registration points, 148–150
resized, performance and, 63
spans, 7
  Director /76 changes, 63
  Director 6/7 changes, 65
terminating before next marker, 76
text, editing on Stage, 378
SRC attribute, 341
stacking windows, 187
Stage, 166–169
  changing, 182–183
  color chip for, 452
  context-sensitive menus, 39, 81
digital video, 538, 557
editing sprite paths and keyframes, 81–85
editing text sprites, 378
exporting, 122–123
hiding/showing, 184
MIAWs
  communications with, 190
  as surrogate Stage, 170
  moving aside (D6), 35
  platform differences, 204
  position on, 148
  aligning cast members, 150
  coordinate properties, 158–160
  coordinates, 151–163
  positioning, 163–165, 167, 183, 225
  properties of, 168
  rendering frames on, 9–11
  shortcuts for, 83
  size of, 19, 63, 163–165, 167, 183, 225
  sprites not appearing on, 10
titlebar
  setting text of, 177
  Windows OS and, 183
transparency of, simulating, 182
updating, Lingo execution and, 13–14
  as window, 166–168
Stage Location property, 184
stage property, 163, 168, 175, 177
Stage Size property, 184
Stage window
  opening/closing (D7), 42
  stageBottom property, 163, 169, 184
  movies and, 172
  stageBox( ), 155
  stageColor property, 168, 397, 417
  hiding desktop, 228
  movies and, 174
StageHand Xtra, 461
stageLeft property, 163, 169, 184
  movies and, 172
stageRight property, 163, 169, 184
  movies and, 172
stageTop property, 163, 169, 184
  movies and, 172
standard buttons, storage space, 258
standard format (movies, castLibs), 109
Standard Import option, 118, 399
Standard Macintosh option, 222
Standard option (Project creation), 227
Start Menu (Windows), 49
starting film loops (simulation), 61
starting MIAWs, 174
starting Projectors, 218
  automatically, 248–249
  with document files, 250
starting up Director, 18
startMovie event, 189
  changing sprite properties, 143
startTime of sprite property, 563, 568
static numeric calculations, 290
static property, 403
step recording, 67
stepFrame event, 10, 92
stereo sound, 485, 496
stop(), 358, 521
  SWAs and, 521
stopMovie event, 189
stopping film loops (simulation), 61
stopSound(), 530
stopSoundInChan(), 530
stopTime of sprite property, 563, 568
storage space, 252–255
  animation, 61
  bitmaps, 398
  cast members, 252–253, 259
    storage space and, 9
  compressed file formats, 109
  compression (see compression)
  disk capacity, 263
  external castLibs and, 103
  file size determination, 146
  Lingo determination of, 261
  loading onto floppy disk, 294
Projectors, 224
removable media, 28
sounds, 485, 509
sprite channels, 5
sprite storage, 6
tweened frames vs. keyframes, 4
unavailable, checking for, 272–276
Windows system, increasing, 30
streaming media
digital video, 539
performance, 290
purging, 267
Shockwave, 330, 348–349
sound, 490, 491, 517
  data rate, 256, 279, 486
  storage space for, 259, 260
streamName of member property, 213, 521
streamStatus events, 349, 355
stretch of sprite property, 159
stretched bitmap sprites, 63
strings
  comparing
    performance, 289
  platform differences, 214
  internationalization concerns, 215
  manipulating, performance of, 258
  strokeColor property, 403
  strokeWidth property, 403
structures (data), memory requirements, 265
Stub Projectors, 219–221
STUB.DIR file, 220
 stylized text
  color chips, 452
  performance and, 36
  substituteFont(), 389
Subtract ink effect, 449
Subtract Pin ink effect, 448
super palette, 424
SW (see Shockwave)
SWA (Shockwave audio), 486, 516–522
  compression, 516–520
  decompression, 519
  Lingo for, 520–522
  playback, 494
  storage space and compression, 208, 261
  troubleshooting, 535
  (see also Shockwave)
SWA Xtras, 297, 298, 310, 313, 321–322
  SWA Compression Xtra, 321
  SWA Decompression Xtra, 528
  SWA Streaming Xtra, 528
swap files (see virtual memory)
swap space, 204
SwapFileMeg option, 255
SWATOMator, 518
Switch Colors command, 417, 441, 450
Switch colors tool (Paint window), 444
Switch ink effect, 446
switchColorDepth property, 225, 397
swLiveConnect attribute, 342
symbols in text, 375–376
symmetry of Score, 18
“Sync to Soundtrack” option, 558, 566
synchronizing media
digital video, 558, 566–568
lip sync, 72
sound cue points, 507, 510–516
Tempo channel for, 5
system configuration, 29–33
cross-platform development, 198
internationalization and, 215
system cursor, 464, 470
system files, Windows OS, 49
System folder (Macintosh), 51
System folder (Windows), 49
system palettes, 423
systemDate( ), 215, 216

T
Tab To Next Field option (Field Properties), 379
tabCount of member property, 384
tabs of member property, 384
Taskbar (Windows), 49
Taylor, John, 308
tell command, 91, 190
tell the stage command, 171
tellStreamStatus( ), 273, 356
Telos AudioActive equipment, 522
tempo (see frame rate)
Tempo channel
  Director 6/7 changes, 66
  sound timing, 510
Tempo option (Tempo channel), 71
terminating sprites before next marker, 76
testing
  asset-specific properties, 134
  beta versions, 287
  compatibility testing, 31–32
delivery testing, 27
digital video is playing, 571
for DV software, 543–546
environment for, 26–29
memory usage, 285
performance, 288
platforms and operating systems, 196
Shockwave, 334
Shockwave movies, 332–335
Shockwave version, 338
sound in Cast, 70
testSpeed( ) (example), 288
text, 369–394
  anomalies, 392–394
  appearance and attributes, 370–376
  attributes for, 379–380
  editing sprites on Stage, 378
  fields (see fields)
  fonts (see fonts)
  formatting, 372–376, 393
  highlighting, 389–391
  hypertext (see hypertext)
  Lingo for, 380–394
    chunk expressions, 385–387
    hypertext-related, 389–392
  modifying in interface, 376–380
  operations on, performance, 289
  registration points, 149, 150
  rich text (see rich text)
  scrolling, 379
  storage space, 259
  stylized/colorized
color chips, 452
  performance and, 36
titlebars (see titlebar)
types of, advantages, 370
  zooming MIAWs and, 185
text fields, importing, 115
text files, linking to, 124
Text Inspector, 38, 376, 459
color chips, 452
text of member property, 382, 385, 461
Text Properties dialog box, 380
  text property, 386
  Text tool (Paint window), 439
Text window
  button shortcuts, 45
textAlign of member property, 382
TextCruncher Xtra, 303, 373
TEXTFOCUS attribute, 342
textFont of member property, 382
textHeight of member property, 382
textSize of member property, 382
textStyle of member property, 382
Index 623

the in Lingo code, xiii
the movieXtraList property, 232
“The requested object could not be found on the server” error, 359
then clause (see if...then statements)
third-party Xtras, 297, 299, 301, 314
graphics formats, 398
platform dependence, 199, 207
32-bit palettes, 411
32-bit Windows differences, 201
thrashing, 290
3D Dreams, 580
throughput, 255, 262, 399
thumbnail for cast members, 66
thumbnail of member property, 445
Thumbnail Size property (Cast), 111
ticks (time), platform differences, 215
ticks property, 232
time (see date and time)
timeoutKeyDown property, movies and, 173
timeoutLapsed property, 173
timeoutMouse property, 173
timeoutPlay property, 173
timeoutScript property, 173
timer property, 232
movies and, 173
timer resolution, 215
timeScale of member property, 215, 555, 563, 570
timeScale of sprite property, 563
timing (see synchronizing media)
tips and tricks
  bundling Xtras into Projectors, 324–326
colorDepth property, 416–417
cursors, 466–467, 469
external castLibs, uses/disadvantages, 103–105
importing, 119
internationalization concerns, 215
Lingo performance, 288, 293
memory optimization, 282–286
MIAW anomalies, 191–192
MIAWs, suggested uses for, 170
operating system, 48–54
Paint window, 440–443, 450
palette fixes, 427–428
performance, improving, 288–294
productivity, 34–36
Projector startup, 219
Projectors, 228
property idiosyncracies, 142
Score, 78–88
shopping for Xtras, 303–305
shortcuts (see shortcuts)
sound
  cue points, 513
  mixing, 499
  SWA compression, 519
system configuration, 30
text anomalies, 392–394
user interface design, 456–457
walking the Score, 99–100
when to use Xtra, 299
title of the stage property, 177, 183
title of window property, 177, 178
titlebar
  MIAWs
    hiding/showing, 178
    setting text of, 177
Stage
  setting text of, 177
  Windows OS and, 183
  windows, modifying buttons on, 180
titleVisible of the stage property, 183
titleVisible of window property, 178
Tool Palette
  button creation, 458
  color chips for, 451, 453
  context-sensitive menu, 38
  shape creation, 406
Tool Xtras, 297, 298
  adding to Xtras menu, 299
Toolbar, context-sensitive menu, 38
Trace Edges tool (Paint window), 444
trace property, 289
Trace Log File property, 242
traceLogFile property, 242, 273
traceLoad property, 273, 274, 289
  Director vs. Projectors, 242
top of rect property, 151, 162, 563
top of sprite property, 148, 152, 159, 563
topSpacing of member property, 384
topSpacing of sprite property, 563
topSpacing of window property, 440
Track Edges tool (Paint window), 444
tTotal Memory (Memory Inspector), 271
Total Used (Memory Inspector), 271
trace property, 289
  Director vs. Projectors, 242
  traceLogFile property, 242, 273
  trackCount( ), 565, 569
Index

trackEnabled(), 565, 569
trackNextKeyTime(), 570
trackPreviousSampleTime(), 570
trackStartStopTime(), 570
trackText(), 565, 569
trackType(), 565, 569
trademark symbol, 376
trails, performance and, 292
trails of sprite property, 10, 407, 563
transferring files across networks, 28
Transform Bitmap command, 442
Transform menu options, 59
transformation properties, 70
transforming palettes, 423
Transition channel, 6, 72–74
Transition Xtras, 297, 298, 323
missing, handling, 315
platform dependence, 207
transitions, 6, 72–74
between palettes, 430
cast members for, 97
changing in Effects channels, 69
digital video and, 551
MIAWs, 192
platform differences, 207
storage space for, 259
transitionType of member property, 74
translation of member property, 563
translation of sprite property, 563
transparency
palettes and, 420
Stage window (simulating), 182
Transparent ink effect, 445, 447
TranZtions Xtra, 73
TreviMedia, 302
triggering sounds (see sound)
troubleshooting
animation techniques, 61
cast member problems, 128
colors and palettes, 425–430
cursors, 466–467, 469
debugging (see debugging)
digital video, 573–578
go command and subsequent commands, 90
hardware/software problems, 23, 202
importing and linked file problems, 120–121
label name errors, 77
media and disk access, 290
memory, 282–286
MIAW anomalies, 191–192
monitor color depth, 412–417
network errors, 359–361
overflow conditions, platform, 213
Paint window, 450
platform (see cross-platform differences)
Projectors, 228
property idiosyncracies, 142
puppeting, 15
Score, 98–100
sound, 533–535
cue points, 513
fades, 526
input sources, 498
sprites not appearing on Stage, 10
testing environment, 26–29
text anomalies, 392–394
uploading Shocked files, 343
X for cast member, 11
Xtras loading and registration, 315–317
(see also tips and tricks)
Tweak window
context-sensitive menu, 38
tweened of sprite property, 95
tweening, 4
Director 6/7 changes, 65
editing sprite paths, 81–85
number of intermediate steps, 62
storage space for, 4
256-color palettes, 204
TYPE attribute, 342
type of member property, 74, 401, 563
castType vs., 142
media types by, 115–121
testing asset-specific properties, 134
type of sprite property, 95, 403
U
unavailable memory, checking for, 272–276
uncompressed files (see compression)
union( ), 162
units, choosing, 164
Unix support for Director, 200
unlinked cast members, 102
unlinked castLibs, 103
      copying cast members, 128
      external, 105
unlinked media types, 115–121
unload command, 268, 510
Unload option, 559
unloadCast command, 269
unloading (see loading)
unLoadMember command, 268, 269, 285
unLoadMovie command, 269
unpuppeting (see puppeting)
unused cast members, deleting, 124
Update Movies (Xtras menu), 298
updateFrame command, 95
updateLock property, 3, 13, 95
updateMovieEnabled property, 96
      Director vs. Projectors, 242
updateStage command, 3, 13
      redundant, 290
updateStage event, 9
UpdateStage web site, xxii
updatering
      digital video, 578
      fields, 393
      Score references to external castLibs,
        108
      Stage, Lingo execution and, 13–14
              (see also redrawing)
upgrading Director, 32
uploading Shocked files to web server,
        342–345
uppercase characters, platform
differences, 210, 214
“url not found” error, 344
url of member property, 213, 521
URLs for source movies, 177
“Usage (Not Used in Score)” option,
        124
Use Hypertext Styles option, 380
Use Media as Available option, 348
Use Movie Settings option, 225
Use System Player option, 227
Use System Temporary Memory option,
        228, 253, 271
Use System Temporary Memory
      Projector creation option, 203
useAlpha of member property, 293,
        445, 447
Used by Program (Memory Inspector),
        271
useFastQuads property, 293, 294
useHyperlinkStyles of member
      property, 392
useHypertextStyles of member
      property, 384, 392
useQuickTimeStreaming( ), 566
user events
delay command and, 92
      Lingo interference with, 14
      sprite properties changes after,
        143–144
      Tempo frame rate and, 5
      waiting for, 71, 93
              (see also runtime; interactivity)
user interface, 456–484
      buttons (see buttons)
cursors (see cursors)
dialog boxes, 482–484
digital video and, 551–566
      menus (see menus)
Paint window options, 435
paletted options, 429
platform differences, 209–211
references on, 457
sounds, 488, 508–510
text manipulation, 376–380
      widgets, 461
userName property, 231, 242

V
V12 Database Engine, 302
value( ), performance, 289
variables, disposing of, 266
      (see also Flash files)
vector graphics, 398, 401
vector shapes, 402–406
      performance, 293
      storage space, 258
version (global variable), 236
version of Shockwave, checking if
      adequate, 338
version property, 353
versions
QuickTime, 541–543
QT3 Pro vs. QT3, 549
Shockwave
version 7 improvements, xix
Xtras, 316
versions of Director
Creator Codes, 32
“Director 5 Style Score display” option, 65
Director 7 bugs (see bugs)
features of Director 7, xvi–xx
Score improvements, 63–69
Shared Cast, 106–107
upgrading, 32
working with multiple, 31–33
(see also Macintosh; Windows operating systems)
vertexList of member property, 403, 404, 405
vertexList property, 405
vertices of vector shapes, 404–406
VFW (Video for Windows), detecting, 543
video (see digital video)
video (see movies)
Video CD, 579
Video for Windows (VFW), detecting, 543
video of member property, 559, 570, 574
video property, 563
Video window (see Digital Video window)
View menu, 42
viewH property, 403
viewPoint property, 403
viewScale property, 403
viewV property, 403
virtual memory, 253
Projectors and, 228
visibility of sprite property, 15, 525
visible of member property, 574
visible of sprite property, 9, 10
movies and, 172
visible of window property, 184
voice narration, 216
volume of member property, 521, 525
volume of sound property, 525, 526

volume of sprite property, 525
volume property, 563
volume, sound, 524–526
volumeLevel of sprite property, 525
volumeLevel property, 563
VREnableHotSpot( ), 566
VRFieldOfView command, 564
VRGetHotSpotRect( ), 566
VRHotSpotEnterCallback command, 564
VRHotSpotExitCallback command, 564
VRML language, 579
VRMotionQuality command, 564
VRMovedCallback command, 564
VRNode command, 564
VRNodeEnterCallback command, 564
VRNodeExitCallback command, 564
VRNodeType command, 564
VRNudge( ), 566
VRPan command, 564
VRPToHotSpotID( ), 566
VRStaticQuality command, 564
VRswing( ), 566
VRTilt command, 564
VRWarpMode command, 565
VSnap, 573

W
Wait cursor, 462
“Wait For All Media” option, 348
“Wait for Cue Point” option, 510, 515
“Wait for End of Digital Video” option, 510
“Wait for End of Sound” option, 510
wait for mouseclick option, 467
Wait options (Tempo), 5, 71, 509, 512
WaitForNetConnection(), 363
waiting
for digital video, 571
for cue points, 72, 512
for number of seconds, 71
for specified number of ticks, 92
for user events, 71, 93
for sounds, 510–511
walking the Score, 99–100
Warp tool (Paint window), 444
watch cursor, 466
Watches window, 39
WAVE files  
cue points, 512  
(see also sound)  
WaveMix sound mixer, 500, 504  
WaveOut device, 497, 500, 502  
Way Cool Screen Saver Engine, 250  
WDEF resources, 178, 204  
web browsers (see browsers)  
Web Download If Necessary option, 227  
web pages, Shockwave movies on, 339–342  
web servers  
multiuser, 362  
uploading Shocked files to, 342–345  
When Needed setting, 123  
“Where is...?” dialog box, 120  
“Where is movie xxxx?” error, 575  
WhiteBook format, 579  
Widget Wizard, 103, 461  
wgtidets, 461  
WIDTH attribute, 341  
width of member property, 158, 559, 563  
width of rect property, 151, 162  
width of sprite property, 15, 148, 159  
zero or negative value, 11  
WIND resource (Macintosh), 182  
window command, 163, 175  
windowID property, 175  
windowList property, 172, 175  
forgetting windows and, 188  
windowPresent( ), 176, 187  
windows  
focus (active windows), 186, 189  
managing multiple open, 44  
media windows, 45  
movies in (see MIAWs)  
moving, 180  
performance issues, 291  
properties of, 176–187  
referring to, 175  
shortcuts for, 42–44  
size and position, 183–186  
Stage as, 166–168  
titlebar  
modifying buttons on, 180  
(see also specific window)  
Windows OS, 195–216  
AutoRun, 249  
Border Xtra with, 179  
CD-ROM, locating, 244–246  
cross-platform strategy, 195–199  
cursors, 462, 469  
date and time, 215  
determining platform, 234–238  
developer shortcuts and tips, 49–51  
development and delivery, 27  
differences from Macintosh, 199–216  
digital video  
cross-platform issues, 548  
detecting software for, 543  
QuickTime installation, 549  
QuickTime support, 541–543  
troubleshooting, 575  
hardware and software requirements, 26  
icon customization, 247  
importing media, 112–115  
limitations to, 199  
math operations, 213  
media file formats, 207–209  
memory and performance, 203, 255  
menus, 210  
monitors, 206–207  
palette problems, 425  
porting checklist, 198  
Projectors, 217, 223, 228  
associating documents with, 251  
Projectors and runtime issues, 202  
screen savers, 232, 249  
Shockwave, 322, 332  
installation (Win98), 335  
shortcuts, 36  
shutDown, restart commands, 229  
sound  
mixing, 496–507  
runtime sound card detection, 530  
sound channels, 489, 496  
Stage, changing, 183  
starting desired Direction version, 32  
string comparisons, 214  
system configuration, 30  
system palette, 423  
transitions, 207  
Windows 3.1, 201  
Windows 95/98/NT, 201
Windows OS (continued)
Xtras, 318
Xtras folder, 319, 324
too many files in, 291
Windows Registry file, 50
Windows System folder, 49
WindowShade Control Panel (Macintosh), 53
windowType of window property, 178–182
redrawing MIAWs, 175
Wind-X Xtra, 177
wipes (see transitions)
Word Wrap (Field Properties), 379
wordWrap of member property, 382, 384
work environment, 24

X
X for cast members, troubleshooting, 11
Xaos Tools Trans-X, 73
XMDFglue XObject, 314
XML Parser (D7), 365
Xoglglu16.DLL library, 314
Xoglglu32.DLL library, 314
XObject, 304, 319
  disposing of, 266
  memory leaks, 284
XtraDraw (see DrawXtra Xtra)
XTRAININFO.TXT file, 226, 305–307
XtraList property, 232, 326
xtraList property, 326
Xtras, 296–327
  agent Xtras, 310
  audio tracks of (see sound)
  cache file for, 318
  color depth changes with, 414
  color-related, 454
  detecting installed, 326
  disposing of, 266
  DLLs required, determining, 317
  font-related, 372
  Lingo (scripting) Xtras, 296, 298, 311, 323
    listing, 325, 326
    loading, 18, 314–318
    mailing list for (Xtras-L), xxiii, 302
    memory leaks, 284
MIAWs as, 170, 174
  missing, handling, 315
MIX-related, 297, 298, 308–310, 323
network-related, 308, 322
  obtaining, 301–305
  online resources for, xxii
platform dependence, 199, 318–319
platform limitations, 211
Projectors with, 224, 226, 322–327
  bundling Xtras into Projectors, 203, 226, 324–326
  Director vs. Projectors, 240
  registering, 314–318
reinstalling after Director upgrade, 32
Shockwave and, 344, 345–348
sound-related, 310, 526–530
Sprite Xtras, 297, 298, 311, 323
  loading, 19
storage space, 261
third-party, 297, 299, 301, 314
too many, 291
Transition Xtras, 297, 298, 323
  transition-related, 73
types of, 296
video-related, 573
  when to use, 299
Xtras folder, 319, 324
too many files in, 291
Xtras/Libs folder, 104
Xtras list (Modify>Movie menu), 325
Xtras menu, 42, 298, 321
XtraZone Xtra, 73

Y
Yamaha MIDI Xtra, 531

Z
Zavatone, Alex, 285
Zeus Productions, xxii, 302
zLaunch utility, 229, 283
zlaunch utility, 550
Zoom tool (Paint window), 438
zoomBox( ), 162
zooming, MIAWs on Stage, 185
zoomWindow event, 189
zPrint Xtra, 303
zWinVer Xtra, 231