CONTENTS

Troubleshooting Lingo

Scripts don't always do what you want the first time. The script often has an error in its syntax: possibly a word is misspelled or a small part of the script is missing. Other times the script might work but doesn't produce the expected result.

Mistakes or bugs almost always occur when writing Lingo, so you should allow enough time for debugging when you develop multimedia titles.

As your skill with Lingo increases, you'll probably encounter different types of problems as you master one area but start learning others. However, the basic troubleshooting techniques described in this section are useful for novice and advanced users alike.

The best way to correct a Lingo bug varies from situation to situation. There aren't one or two standard procedures that resolve the problem. You need to use a variety of tools and techniques, such as the following:

- An overview and understanding of how scripts in the movie interact with each other
- Familiarity and practice with common debugging methods

The following tools help you identify problems in Lingo:

- The Message window, when tracing is on, displays a record of the frames that play and the handlers that run in the movie.
- The Debugger window displays the Lingo that is currently running, the sequence of handlers that Lingo ran to get to its current point, and the value of variables and expressions that you select in the script.
- The Watcher window displays the value of variables and expressions that you select in the Script window.
- The Script window lets you enter comments, insert stopping points in the script, and select variables whose value is displayed in the Watcher window.

Good scripting habits

Good scripting habits can help you avoid many Lingo problems in the first place.

- Try to write Lingo in small sets of statements and test each one as you write it. This isolates potential problems where they are easier to identify.
- Insert comments that explain what the Lingo statements are intended to do and what the values in the script are for. This makes it easier to understand the script if you return to it later or if someone else works on it. For example, the comments in these Lingo statements make the purpose of this `if...then` statement and repeat loop clear:

  ```
  if the soundLevel = 0 then
      -- Loop for 4 seconds (240 ticks).
      startTimer
      repeat while the timer < 240
      end repeat
  end if
  ```

- Make sure that Lingo syntax is correct. Use the Script window's Lingo menu to insert preformatted versions of Lingo elements. Rely on the *Lingo Dictionary* to check that statements are set up correctly.
- Use variable names that indicate the variables' purpose. For example, a variable that contains a number should be called something like `newNumber` instead of `ABC`.

Basic debugging

Debugging involves strategy and analysis, not a standard step-by-step procedure. This section describes the basic debugging approaches that programmers successfully use to debug any code, not just Lingo.

Before you modify a movie significantly, always make a backup copy.

Identifying the problem

It might seem obvious, but the first thing to do when debugging is to identify the problem. Is a button doing the wrong thing? Is the movie going to the wrong frame? Is a field not editable when it should be?

If you copied the Lingo from another movie or from a written example, check whether the Lingo was designed for some specific conditions. Perhaps it requires that a sprite channel is already puppetted. Maybe cast member names must follow a specific style convention.

Locating the problem

Do the following to start locating a problem:

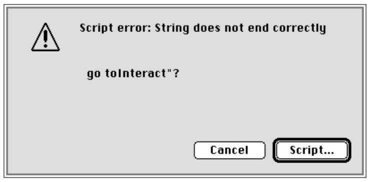
- Think backwards through the chain to identify where the unexpected started to happen.
- Use the Message window to trace which frames the movie goes through and the handlers that Lingo runs.
- Determine what the Lingo should be doing and consider what in these statements relates to the problem. For example, if a text cast member isn't editable when you expect it to be, where in the movie does (or doesn't) Lingo set the cast member's `editable` property?
- If a sprite doesn't change as intended on the Stage, is the `updateStage` command needed somewhere?
- Does the problem occur only on certain computers and not others? Does it happen only when the display is set to millions of colors? Maybe something in the computer is interfering with the application.

You can focus on specific lines of Lingo by inserting a breakpoint—a point where Lingo pauses—in a line. This gives you a chance to analyze conditions at that point before Lingo proceeds. For information on how to insert breakpoints in a script, see Using the Debugger window.

Solving simple problems

When finding a bug, it's a good idea to check for simple problems first.

The first debugging test occurs when you close the Script window. Director gives you an error message if the script contains incorrect syntax when you close the Script window. The message usually includes the line in which the problem was first detected. A question mark appears at the spot in the line where Director first found the problem.



```
Script error: String does not end correctly

go toInteract"?
```

For example, the message "String does not end correctly" tells you that the problem relates to the string in the line `go to Interact`. The only string in this line is Interact, which is the name of a marker. After thinking about syntax for strings, you'll see that the problem is that no beginning quotation mark precedes Interact.

Syntax errors are probably the most common bug in Lingo. When a script fails, it is a good idea to first make sure that:

- Terms are spelled correctly, spaces are in the correct places, and necessary punctuation is used. Lingo can't interpret incorrect syntax.
- Quotation marks surround the names of cast members, labels, and strings within the statement.
- All necessary parameters are present. The specific parameters depend on the individual element. See the *Lingo Dictionary* to determine any additional parameters an element requires.

If the Script window closes without an error message, the script might contain a bug. If Lingo isn't doing what you want, check that:

- Values for parameters are correct. For example, using an incorrect value for the number of beeps that you want the `beep` command to generate gives you the wrong number of beeps.
- Values that change—such as variables and the content of text cast members—have the values you want. You can display their values in the Watcher window or in the Message window by using the `put` command.
- The Lingo elements do what you think they do. You can check their behavior by referring to the *Lingo Dictionary*.

Advanced debugging

If the problem isn't easy to identify, try the following approaches:

- Determine which section has the problem. For example, if clicking a button produces the wrong result, investigate the script assigned to the button.

  If a sprite does the wrong thing, try checking the sprite's properties related to the sprite. Are they set to the values you want when you want?
- Figure out where the Lingo flows. When a section of the movie doesn't do what you want, first try to trace the movie's sequence of events in your head. Look at other scripts in the message hierarchy to make sure Director is running the correct handler.
- Follow the tracing in the Message window; this shows which frames the movie goes through and any handlers that the movie calls as the movie plays.
- Try using the Step Script and Step Into features in the Debugger window and see whether the results differ from what you expect.

- Check variables and expressions. Analyze how their values change as the movie plays. See if they change at the wrong time or don't change at all. If the same variable is used in more than one handler, make sure that each handler that uses the variable states that the variable is global.

  You can track the values of variables and expressions by displaying their values in the Watcher window.

- Make changes one at a time. Don't be afraid to change things in a handler to see if the change eliminates the problem or gives some result that helps point to the problem.

  However, don't trade one problem for another. Change things one at a time and change them back if the problem isn't fixed. If you introduce too many changes before solving a problem, you might not determine what the original problem was and you might even introduce new problems.

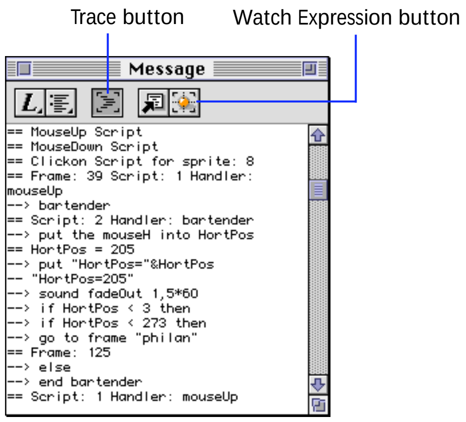
- Re-create the section. If you haven't found the problem, try re-creating the section from scratch. For example, if rolling the cursor over a sprite doesn't make the sprite behave the way you want, create a simple movie that contains just the sprite and handler with the `rollover` statement.

  Don't just copy and paste scripts; that might just copy the problem. Re-creating the section lets you reconstruct the logic at its most basic level and verify that Director is working as you expect. If the section that you re-create still doesn't work properly, chances are that there is something wrong in the logic for the section.

  If the section that you re-create works properly, compare that section to the original movie to see where the two differ. You can also copy the section into the original piece and see whether this corrects the problem.

Using the Message window

When its Trace button is turned on, the Message window displays a record of what the movie does as it plays. This is useful for tracking Lingo's flow and seeing the result of specific lines of Lingo.



```
== MouseUp Script
== MouseDown Script
== Clickon Script for sprite: 8
== Frame: 39 Script: 1 Handler:
mouseUp
--> bartender
== Script: 2 Handler: bartender
--> put the mouseH into HortPos
== HortPos = 205
--> put "HortPos="&HortPos
-- "HortPos=205"
--> sound fadeOut 1,5*60
--> if HortPos < 3 then
--> if HortPos < 273 then
--> go to frame "philan"
== Frame: 125
--> else
--> end bartender
== Script: 1 Handler: mouseUp
```

Entries after a double equal sign (==) indicate what's occurred in the movie—such as which frame the movie has just entered, which script is running, or the result of a function or setting a value. For example, the line:

```
Frame: 39 Script: 1 Handler: mouseUp
```

indicates several things:

- The movie entered frame 39.
- The movie ran script 1.
- The movie ran the `on mouseUp` handler in script 1 after the movie entered the frame.

Entries after an arrow made up of a double hyphen and right angle pointer (-->) are lines of Lingo that ran. For example, the lines:

```
--> sound fadeOut 1,5*60
--> if leftSide < 10 then
--> if leftSide < 200 then
--> go to frame "Game Start"
```

indicate that these Lingo statements ran. Suppose you were trying to determine why the playback head didn't go to the frame labeled "Game Start." If the line `--> go to frame "Game Start"` never appeared in the Message window, maybe the condition in the previous statement wasn't what you expected.

You can also follow how a variable changes by selecting it in the Message window and then clicking the Watch Expression button. Director then adds the variable to the Watcher window, where its value is displayed and updated as the movie plays. For more information about the Watcher window, see the next section “Watcher window.”

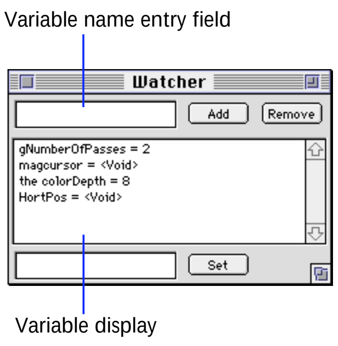
You can test a one-line Lingo statement by typing the statement directly in the Message window and then pressing Enter (Windows) or Return (Macintosh). Lingo executes the statement.

If the statement is valid, the Message window displays the result of the statement. If the script is invalid, an alert appears.

You can test a handler by writing it in a Movie Script or Score Script window and then calling it from the Message window. To do this, type the handler name in the Message window and then press Enter (Windows) or Return (Macintosh).

Using the Watcher window

You can check the values of variables and expressions while the movie plays by displaying the values in the Watcher window.



```
gNumberOfPasses = 2
magcursor = <Void>
the colorDepth = 8
HortPos = <Void>
```

The variable or expression appears in the variable display followed by an equal sign (=) and the expression or variable’s current value. If the value of an expression or variable isn’t available, the term <Void> appears to the right of the equal sign. Director updates values in the Watcher window when you step through lines of a script while in debug mode or when the movie pauses.

To add variables or expressions to the list in the Watcher window:

1 Open a Script window in which the variable or expression appears.

2 Select the variable or expression.

3 Click the Watch Expression button at the top of the Script window.

Using the Debugger window

The Debugger window can help you locate and correct bugs in Lingo scripts. It includes several tools that let you do the following:

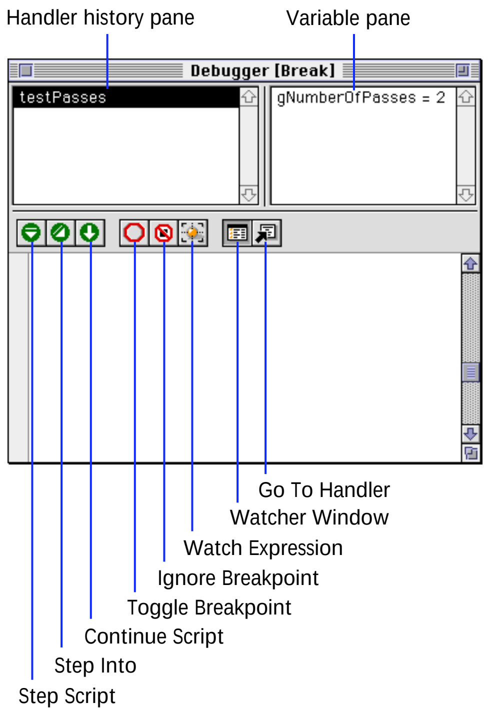
- See the part of the script that includes the current line of Lingo
- Track the sequence of handlers that were called before getting to the current handler
- Run selected parts of the current handler
- Display the value of any local variable, global variable, or property related to the Lingo that you're investigating
- Open related windows such as the Watcher window and Script window

To open the Debugger window, do one of the following:

- Choose Window > Debugger
- Click Debugger window in the Lingo error alert box that appears when Director encounters a run time error in a script

The Debugger window also opens when Director encounters a breakpoint in a script.

When the Debugger window opens, it shows the current line of Lingo and offers several choices for what to run next.



To see which is the current line of Lingo:

1 Check the script pane.

   The script pane displays the current script and highlights the current line of Lingo.

2 Find the line that has a green arrow next to it in the left margin.

   The green arrow points to the current line. You can't select a different line of Lingo by clicking it in the script pane.

You can't edit the script directly in the Debugger window; you must return to the Script window to edit a script.

To see the sequence of handlers that Lingo ran before getting to the current point:

Check the list of handlers in the handler history pane.

The handler history pane displays the sequence of handlers that the movie has run leading up to the current handler. Clicking a handler name displays the variables that are in that handler.

To see the values for variables and expressions used in Lingo that the movie has just run:

Check their values in the variable pane.

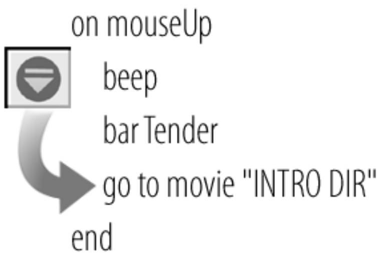
The variable pane displays local variables, global variables, and property settings that Lingo encountered as it ran up to the current break point. You can't change these values from the Debugger window. To change any of these values while working with the Debugger window, use a set or put statement in the Message window.

Many handlers include calling statements to other handlers. You can focus your attention on such nested handlers or ignore them and focus on Lingo in the current handler.

When you are confident that nested handlers are performing as expected and want to concentrate on Lingo in the current handler, the Debugger window can step over nested handlers and go directly to the next line of Lingo in the current handler.

To step over nested handlers:

Click the Step Script button in the Debugger window.



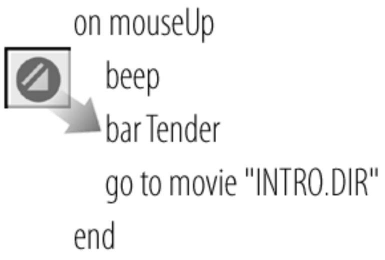
```
on mouseUp
  beep
  barTender
  go to movie "INTRO DIR"
end
```

The Step Script button runs the current line of Lingo, runs any nested handlers that the line calls, and then stops at the next line in the handler.

If you suspect that nested handlers aren't performing as expected and want to study their behavior, the Debugger window can run nested handlers in the same order that Lingo would execute them in a normal movie.

To run nested handlers:

Click the Step Into button in the Debugger window.



```
on mouseUp
   beep
   bar Tender
   go to movie "INTRO.DIR"
end
```

Clicking the Step Into button runs the current line of Lingo and follows Lingo's normal flow through any nested handlers called by that line. When finished with the nested handlers, the Debugger window stops at the next line of Lingo within the nested handler.

Using the Step Script or Step Into button always puts the Debugger window on the bottommost active handler in the pane.

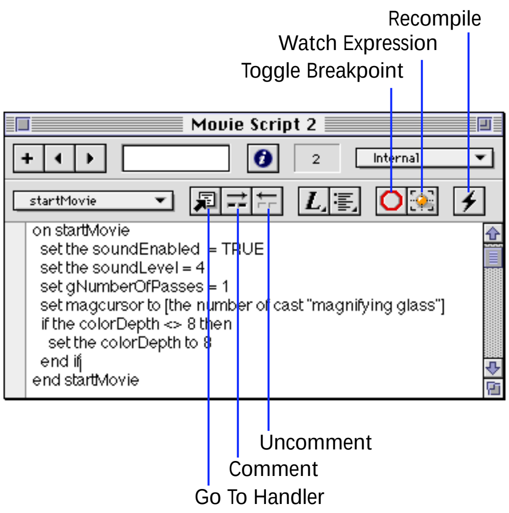
To leave the Debugger window:

Click the Continue button or the Go To Handler button.

- Clicking Continue exits debugging and returns to the movie.
- Clicking Go To Handler returns to the Script window and goes to the handler that is currently highlighted in the script pane.

Using the Script window

Several buttons in the Script window help debug scripts.



```
on startMovie
  set the soundEnabled = TRUE
  set the soundLevel = 4
  set gNumberOfPasses = 1
  set magcursor to [the number of cast "magnifying glass"]
  if the colorDepth <> 8 then
    set the colorDepth to 8
  end if
end startMovie
```

Use these buttons to do the following:

To go to the handler selected in the Script window.

Click Go To Handler

To make the current line of Lingo a comment.

Click Comment.

To remove commenting from the current line of Lingo.

Click Uncomment.

To turn breakpoints in the current line of Lingo on and off.

Click Toggle Breakpoint.

To add the selected expression or variable to the Watcher window.

Click Watch Expression.

To recompile the movie’s scripts.

Click Recompile.

INDEX