Debugging Hints

BEFORE WRITING YOUR SCRIPTS

• Use any resources you have to research the tasks involved in your application. Keep a list of Lingo terms that might be useful.
• Make sure you understand the simpler concepts described before moving on to reading about the more advanced possibilities. If necessary, borrow a book that is more geared towards beginners in order to master the foundational concepts. Often times reading multiple descriptions of the same thing is necessary for thorough comprehension of a difficult concept.
• If you are using *Lingo in a Nutshell*, pay special attention to the lists of common scripting errors described in each chapter.
• Using Director’s built-in Lingo Dictionary under the Help menu, look up any terms that you think might be useful. Investigate the “see also” Lingo terms.
• Think through the different steps and establish what custom handlers and variables you might need. Some people find it helpful to sketch code and diagrams out on paper first.
• Try out expressions in the message window to see if you get the results you expect.
• Isolate the most challenging part of your scripting task, and see if you can come up with a simplified scenario with which to experiment before embarking on the real thing.
• Think about ways to make things easier for yourself. For instance, name your cast members with a text string that works with your code.

THROUGHOUT THE SCRIPTING PROCESS

• If you find yourself accidentally using the cast member number when you meant to use the sprite number, name your cast members so that the name and not the number of the cast member appears on the score.
• Save multiple versions of your file so that you can always go back to a version that was completed before the scripting problems started to arise.
• Comment your scripts while you still remember something about what you were doing. Such comments can be useful during the scripting process as well as much later when you are trying to re-purpose the code for another project.
• Use descriptive names for variables, casts, cast members, and custom handlers in order to avoid confusion. Be specific and consistent in your terminology. Avoid reserved Lingo terms.
• Use the pop-up menus in the scripting window in order to avoid typos and to be reminded of the proper syntax.
• While advanced coders may think of the simplest and most elegant way to solve a problem before beginning the scripting process, novices should aim for functionality first and elegance second.
• If you “hardcode” various values into your scripts, be careful when you move around sprites and cast members. You could be interfering with the functionality of your scripts.
• Once you have functionality with hardcoded scripts, go back and try to eliminate the hardcoded values.
• If you find yourself writing the same or a similar script multiple times, think of ways to generalize the script for reusability. For instance, use “sprite (the currentSpriteNum)” in sprite scripts in order to make them generalizable.
WHEN YOU GET SCRIPTING ERRORS

• If you decide to try a different approach after getting a scripting error, instead of deleting the troubled script, use the comment/uncomment function in the scripting window so that you can temporarily deactivate the script and return to it later.
• Read error messages carefully and look for consistent mistakes. For instance, do you often leave out the “the” when referring to properties? Then you might get an error telling you that a variable is not defined. A similar message might come up if you forget to declare a global variable. Use Appendix E of *Lingo in a Nutshell* to learn more about what individual error messages mean.
• Learn to isolate the problem by breaking a script down into parts that you can test individually.
• Turn on the trace function in the message window while your movie is playing in order to see what scripts are enacted in what order and when certain error pops up.
• If you find the trace to be overwhelming, use selected “put” statements in your script. Use concatenation of text strings to produce a meaningful sentence in the message window.
  
  ```
  set randomDollarAmount = random(3000)
  -- A random number between 1 and 3000 is chosen to be the amount of money
  -- available to the user.
  put "The amount of money in my wallet is $" & randomDollarAmount
  -- The message window will tell you that amount in a coherent statement.
  ```
• If it seems that new changes to a script are not being registered, hit the lightning button in the script window to recompile an individual script or try Control > Recompile All Scripts.
• Set a breakpoint in a script by clicking in the margin of the scripting window next to a particular expression. A red dot should appear next to the expression. During playback, the Debugger will open up at that particular point to allow you view the pertinent handlers and variables at the time a line of code is enacted.
• If your problem seems to center around a particular list or variable, highlight the item in the scripting window and click on the “watch expression” button. The message window will keep track of the changing values of the list or variable during runtime.
• Use the message window during runtime to check on variables. Simply type “showGlobals” or “showLocals” in order to see a list of variables at any point in the movie.
• Check the coweb to see if other people have found solutions to similar problems.
• Walk away and return to the problem at a later date when your mind is clearer.
• If you solve a problem that you think others might run into, document it on the coweb.