- **Week 5, day 1—Lingo scripting intro.**
  - **How to program: the executive summary**
    - **What is a program**
      - List of actions to be executed automatically.
      - The order in which the various actions are taken, and the specifics of their operation, may change at *runtime* (each time the program is fed through the computer).
    - **Basic parts of a program**
      - *Containers*
        - Storage units for the items (text, numbers, pictures, etc.) that the program works with.
        - Containers common to most languages: constants, variables, files.
        - Scripting languages such as Lingo and HyperTalk add special containers such as fields. They often lack other types of containers such as arrays and records. They also lack *typing*, a way to prevent mistakes by accepting only specific types of data for each container.
        - Most scripting languages extend the notion of containers to *objects* (ex.: buttons), which combine data items (ex.: the picture of the button) and programs (ex.: what happens when the button is clicked).
      - *Statements*
        - The actions to be taken when the program runs
          - *Commands*
            - A predefined set of operations that the computer can perform
          - *Parameters*
            - Specify how the command should be carried out, and/or what it should affect
      - *Flow control*
        - Pertains to the order in which each part of the script will execute
          - *Conditional tests*
          - *Loops*
          - *Jumps* and *subprograms*
      - *Comments*
        - Text that is only for human consumption—not a command to be executed
        - Comments are *very* important in figuring out the logic behind a script written by someone else, or one you wrote some time ago. Most programming is maintenance/modification of existing code.
    - **Programming process**
      - Define *specifications* (what the program should do, and on which items)
      - Organize the items that the program will work on (*data structures*)
      - Break the specs into simple steps (*algorithm*)
      - Lookup *commands* that will do each step
      - Determine appropriate *parameters*
      - Test and debug
        - Execute program lines selectively (*step, comment/uncomment*)
        - Stop program to examine its state (*breakpoints*)
        - Display parts of the program as they execute (*trace*)
        - Display variable values at key moments in the program to ensure they are valid
    - **Scripting languages vs. "monolythic" languages**

- In languages such as Basic or Pascal, the program is generally a single text file which the computer scans from top to bottom to gather instructions. In scripting languages such as Lingo and HyperTalk, the text of the program is scattered in a number of locations (*scripts*), of which only a fraction may be scanned each time the program runs.
- *Scope* (how far the influence of a portion of the program reaches)
  - In scripting languages, the scope of each portion of the program and its order of execution are determined by which script it resides in.
- *Message passing* (the way that various parts of the program are activated)
  - In scripting languages, there is no single starting point for the program. Various parts can be activated at different times based on the *events* registered on the computer as the program runs.
- **3-step process**
  - Determine event that will trigger each part of the program (*message*)
  - Determine where to place each part of the program (*script hierarchy*)
  - Follow the standard programming process to respond to the message (*message handler*)

- **Director intro**
  - **Overview and comparison to HyperCard**
    - Director has two completely different states:
      - when the movie is stopped (in which case you can author)
      - when the movie is playing (when the effects of authoring become apparent)
      - This is somewhat similar to the authoring mode of HyperCard (when the field, button, or painting tools are selected) vs. the browsing mode (when the browser tool is selected)
    - Frames are seldom used the way cards are used:
      - A HyeperCard card is generally a complete display screen
      - Director display screens are often spread over several frames to allow for animating elements
      - In HyperCard as in Director, only one card or frame is visible at any one time from each stack or movie
    - Better-suited than HyperCard to graphics-intensive entertainment applications.
      - Animation, color graphics fully integrated and easily implemented
      - Supports basic vector graphics
    - Less-suited than HyperCard to information-intensive applications.
      - Hypertext facilities extremely primitive.
      - Linear playback bias gets in the way of building complex webs of links.
  - **Navigation**
    - Cmd-R to rewind
      - Useful even for a one-frame movie, as it resets moveable sprites to their initial position.
    - Cmd-P to play (or use the Enter key on the keypad)
    - Cmd-period to stop
    - Cmd-1 to toggle between foregrounding Stage and foregrounding all other windows
    - Shift-Enter to foreground Stage, play movie, then background Stage
    - Cmd-left/right arrow keys to go to prev/next frame
  - **Online help**
    - Shift-option select a window or menu command
    - no balloon help
  - **Windows**

- Stage
  - Stage coordinates increase from top to bottom and from left to right
- Cast
  - Icon (lower-left of thumbnail) indicates type of cast member.
  - Cast-->Cast Window Options: select size of cast members' thumbnails
- Score
  - Each column is a Frame (a group of elements displayed at one time)
  - Each row is a Channel (a location for one display element)
    - 6 special-purpose channels (at the top of the Score)
      - Tempo
      - Palette
      - Transition
      - 2 Sound channels
      - Script
    - 48 sprite channels for graphic elements
      - an object in a higher-number channel will appear in front of objects in lower-number channels
  - The intersection of a frame and a channel is a Cell

- **Media elements**
  - Castmembers
    - Any element imported/created in Director for use in a movie (sounds, graphics, text, etc.)
  - Sprites
    - A particular instance of a castmember. Different sprites based on the same Cast member can have different attributes (position, size, etc.)

- **The mechanics of Director scripting**
  - **Kinds of scripts**
    - immediate execution in Message window
      - use to test fragments of code
    - Primary event handlers
      - Available at any time or place in the movie (default behavior for 4 circumstances):
        - keyDownScript
        - mouseDownScript
        - mouseUpScript
        - timeOutScript
      - Define and undefine as needed in any of the other scripts
    - Cast Members' scripts
      - entered by selecting the cast member's thumbnail in the Cast window and clicking the Script button.
      - Any cast member (bitmap, PICT, text, shape, button) can have a script attached to it.
      - Cast-->Cast Window Options: check "Indicate Cast Members with Scripts" to have an L-shaped indicator appear in the lower-left corner of cast members' thumbnails when a script is attached to them.
      - responds to mouse events addressed to sprites based on the cast member. Use to define default effects of clicking on cast member's sprites
    - Script Cast Members
      - Entered in the Script window (display using Window menu)
        - Click the Plus button to add a new script
        - Click the Arrow buttons to review existing scripts

        - Click Info (“i”) button to choose between Movie and Score type
    - Movie scripts
        - Cast member icon has black bar on the right
        - use for event scripts that handle events outside of castmembers
        - use as a repository for scripts that must be available to Score and/or Cast scripts
    - Score scripts
        - Also entered by selecting a cell in the Score window, then choosing New from the Script pop-up menu
        - Appear in the Cast (cast member icon has no black bar). Also listed in the Score script pop-up.
        - Sprite scripts
            - respond only to mouse and keyboard events addressed to the sprite.
            - use to override, for that one cell, the default action established in the script of the corresponding cast member. Different sprites based on the same cast member may have different scripts.
        - Frame scripts
            - placed in script channel of Score window
            - use for simple frame-specific flow control: define what happens when movie enters or exits the frame
- **Writing scripts**
    - Message window (Cmd-M)
        - put insertion pt at the end of any line and press Return to execute the statement on that line
    - Script preview area of Score window
        - click to open script window and edit script
    - Double-click script cast member
    - Script controls/display in the Score window
        - Script pop-up menu
            - Script numbers same as Cast member position
        - Script notation (Display pop-up menu)
        - Frame markers and labels
        - Removing script from the Score
    - Editing features
        - Script editing window
            - Automatic formatting (press Tab key)
                - Indentation indicates wether syntax is correct
            - Save, recompile and exit (press Enter key on the keypad)
        - Lingo menu
            - Select a command to have a prototype of its syntax inserted at the insertion pt.
        - Text menu
            - Search and replace
            - Comment/Uncomment
            - Recompile script (checks correctness beyond Tab key)
        - Replacing scripts vs. creating new ones
            - when pulling down the the Script pop-up menu in the Score window, choosing New allows entering a wholly new script for the cell, while selecting one of the numbered scripts will attach it to the cell
    - Writing style
        - when referring to names of frame labels (even those that contain no spaces) always enclose in quotes (unlike object names in HC)

- Lingo is case-insensitive, but consistent capitalization helps understand program:
  - commands, functions, keywords, variables, methods are all lower case, except for compound words
  - each part of a compound word (except the first) is capitalized—improves readability in spite of the lack of spaces.
  - Lingo constants are all-caps
- Indentation and blank lines highlight program structure
- Any alphanumeric combination starting with a letter is an acceptable name for variables and handlers, but consistent and descriptive names helps understand program:
  - Precede global variable names with lower-case ‘g’
  - Precede method names with lower-case ‘m’
  - Check that your names are not the same as one of Lingo’s predefined names.
- Refer to castmembers by name, and to frames by labels
  - note: you must refer to a castmember by number when changing the castmember associated with a sprite
    - use the number of cast "castName" to arrive at the cast number when the cast name is known.
- Provide exhaustive comments
  - comments are any text starting with -- (two hyphens) and ending with a carriage return.
  - Comments on a separate line are generally more readable than in-line comments
  - For handlers, place comment line right after the ‘on handlerName’ line
  - Control structures (e.g., loops) should have a comment line right after the ‘end’ line—it will be automatically indented for best legibility.