Scripting – The Anatomy of Lingo

- OOP – Object-Oriented Programming (pp. 245-248)
- events & handlers
    - User Feedback events: keyDown, keyUp, mouseDown, mouseUp, mouseEnter, mouseLeave, mouseWithin, mouseUpOutside
    - Playback events:
- Lingo Elements
    - commands – tell a movie to do something (e.g., halt, go to, etc.)
    - functions – return a value (e.g., date)
    - keywords – reserved words that have special meaning in Lingo (cannot be used as variable names)
    - properties – attributes of an object (e.g., visible, stageColor, etc.)
    - operators – terms that change or compare one or more values (e.g., >, <, =, +, etc.)
    - constants – elements that represent an unchanging value (e.g., TRUE, FALSE)

The Message Window

Allows you to track messages sent whenever an event occurs in Director

- Open the Message Window (Window→Message)
- Type the following text into the Message Window:
  ```
  put the date
  ```
- Press the Enter/Return key
    - notice how the date appears on the line below your typed command
- Now try the following:
  ```
  put the time
  ```
    - cool, eh? … you can monitor actions in the message window while your movie is running or you can pause it at any time to check the value of one of your variables
    - you can also use the **`put`** command in your Lingo scripts to have values typed automatically into the Message Window as your movie continues to play
        - Create a new movie (File→New→Movie)
        - double-click on frame 1 of the Frame channel and type the following text into the Script Window

```
on exitFrame me
  put the time
  go to the frame
end
```

        - Run the movie and monitor the output of the Message Window
            - **Challenge**: Can you modify this 1-frame movie so that it outputs the time to the Message Window only every 5 seconds? There are several ways to accomplish this, but can you do so using the Tempo channel?
    - At any time, you can select all of the text in the Message Window and press the Delete key to clear the contents

Tracing in the Message Window

The middle toolbar button in the Message Window allows you to "Trace" all messages being sent while your movie is running. Try the following

- Open any movie you have created that contains scripts and/or behaviors
    - the movie you created in the previous section would be an excellent choice for this demonstration
- Open the Message Window and click on the Trace button (it should appear to be depressed)
- Begin playback of your movie and monitor the contents of the Message Window.
- Stop playback of the movie and scroll through the contents of the text in the Message Window; notice how each scripted command is dutifully recorded for your review
    - This can be an extremely helpful capability as you begin to utilize more and more of your own Lingo scripts

Using the Behavior Inspector to Add Interactivity

The Behavior Inspector is divided into three primary sections: the Behavior pop-up & list, the Event pop-up & list, and the Action pop-up and list. If you do not see all three of these areas, simply click on any right-facing triangles you see and additional options will be made available.

- File→New→Movie
- Create a cast member (go ahead … *you* choose the type)
- Drag the cast member to the center of the stage
- While the cast member has focus, open the Behavior Inspector if it is not already on visible
- Add a New Behavior that will tell your user the date
    - From the Behavior pop-up, select "New Behavior"
        - Provide an appropriate (ShowDate) name in the textbox that appears
    - From the Event pop-up, select "**`mouseUp`**"
    - Look through the options on the Action pop-up and notice what menu & submenus are available to you automatically.
        - Instead of selecting one of these, we are going to create our own command
    - Click on the "Script Window" icon on the Behavior Inspector toolbar
    - In the window that appears, type the following line between the **`on mouseUp`** and **`end`** entries
      `alert ("Today is " & the date)`
    - Run the movie and observe what happens when you click on the sprite
        - Notes:
            - **`alert`** is a command that allows you to communicate a message (whatever is contained in the parentheses after this keyword) to your user
            - the space before the closing quote is important, since without it date would appear right after the "s" in "is"

- the ampersand ("&") is an operator used to merge – concatenate – strings; you could use a double ampersand ("&&") which would add a space before the concatenation

- A complete version of this movie is available as "showDate.dir" on the 337 server in the folder labeled "_Movies"
  - Challenges:
    - Can you figure out a way to add a period to the end of the sentence in the alert box?
    - Can you figure out a way to inform you user of both the date *and* the time?
    - Using the keyword **`halt`**, can you modify this movie by adding a push button that cause playback of your movie to stop?
- A complete version of this movie is available as "showDate2.dir" on the 337 server in the folder labeled "_Movies"

Experiments

create movie with pushbutton that changes stageColor to random color

- File→New→Movie
- Insert→Control→Push Button
- Type a label for the button ("Press Me!")
- Click on the button cast member in the Cast Window (Window→Cast)
  - Notice the toolbar at the top of the Cast Window
    - In the "Cast Member Name" textbox, type "ButtonPress"
    - Click on the "Cast Member Script" icon … just to the right of the "Cast Member Name" textbox; if you do not see the icon, you may need to expand the Cast Window horizontally until it becomes visible
      - alternatively, while the cast member is selected, you could open the Property Inspector, click on the "Member" tab, and click on the "Cast Member Script" icon about half way down on the right side [if you do not see this icon, click on the right-facing triangle at the bottom of the Property Inspector to expand the box, revealing additional options]
- The Script Window will open with a blank script template for the mouseUp event … just the event to which you want your script to respond
  - notice how the script is laid out
    - the first line tells the event that will trigger the script
    - the last line ("end") tells Director that the script is done
    - the blank space in between is where you provide instructions to Director using *Lingo*
- Make sure that the cursor is in the blank space between the "on mouseUp" and "end" lines, then type the following text exactly:

```
on mouseUp
```

```
  the stageColor = random(255)
end
```

- In this Lingo command, you are telling Director to select a random number between 0 and 255 (an 8-bit value), then assign the color located in that location in the current color palette to the stageColor property (a system property that sets the background color of the Stage)
  - **`random()`** is a Director function that automatically selects and returns a value between 0 and the number you type in the parentheses

- Before running the movie, let's add one more script …
  - First make certain that the "Loop Playback" toggle button on the "Control Panel" is set to so that the movie does not loop
  - Double-click on Frame 1 of the Frame Script Channel (the bottom row of the top portion of the Score)
  - In the Script Window that appears, type the one line command below, show the script looks like this:

```
on exitFrame me
  go to the frame
end
```

    - In this command, you are simply telling Director that, when the Playback head exits the current Frame, it should return to the same frame ("the frame")

- Now run the movie and notice that, when you click on the "Press Me!" button, the background color on the Stage changes
  - If your movie starts, then stops immediately, set the "Loop Playback" toggle switch on the "Control Panel" (Window→Control Panel) to the Loop setting
  - As an alternative, add a second script to the

- **`stageColor`** vs. **`bgColor`**
  - **`stageColor`** allows you to set the background color of the stage to many colors (either 0 to 15 for a 4-bit value or 0 to 255 for an 8-bit value)
  - to gain greater control over these colors you can use another one of Lingo's built-in commands:
    `rgb(redValue, greenValue, blueValue)`
    - this method provides you with much more control over the actual color that will appear than the **`stageColor`** property, since it does not really on the user's current color palette … instead, it combines values that you provide for the amount of red, green, and blue to arrive at the appropriate color
  - Open the Cast Member Script for the "Press Me!" button and replace the **`stageColor`** command with the following **`rgb()`** statement provided below:

```
on mouseUp
  (the stage).bgColor = rgb(random(255),random(255),random(255))
end
```

    - This statement sets the bgColor property of the "Stage" object to a value determined by randomly generating a value (from 0 to 255)

for the amount of red, green, and blue that will create the new background color of the Stage

    - Now playback the movie and see if you notice more subtle changes in the colors that appear in response to your mouse clicks; in the unlikely event that your user has her/his system colors set so that only 256 colors are available, you will probably not see any visible difference … however, as mentioned above, the colors created using **`rgb()`** will be more accurate and consistent than those created using **`stageColor()`**
- A complete version of this movie is available as "stageColor.dir" on the 337 server in the folder labeled "_Movies"

Modify "stageColor.dir" to change the `foreColor` & `bgColor` of a text sprite

- Add two new cast members
    - Click on the first empty cast member (number 3) and open the Text Window
        - Type some text (e.g., "Model Text") and make certain to set the margins of your text window so that the total width is not much wider than the characters you just typed
        - Close the Text Window and drag the newly created text member on to the stage just above the "Push Me" button
            - check and make certain that this new sprite appears in channel 2 of the Score Window; the lingo commands you will add below require that the Text Sprite be sprite number 2
    - Click on the next empty cast member (number 4) and add another Push Button (Insert→Control→Push Button)
        - for the label, type "Change Text", then move this second button to the lower right-hand corner of the Stage
- In the Cast Window, click on the Change Text button you just added and type a name in the Cast Window toolbar, then click on the Cast Member Script icon to open the Script Window
    - Type the following …

```
on mouseUp
  sprite(2).foreColor = random(255)
  sprite(2).backColor = random(255)
end
```

        - These Lingo commands set the text (**`foreColor`** property) and the background (**`backColor`** property) to random colors in the current palette
- Now, run the movie and notice that, when you click on the "Press Me!" button, the background color of the Stage changes and, when you click on the "Change Text" button, both the text color and the background color of the Text Sprite change to (usually) different values
- A complete version of this movie is available as "changeTextColor.dir" on the 337 server in the folder labeled "_Movies"

- **Challenge**: figure out a way to set the background color of the Text Sprite using `bgColor` and the `rgb()` method instead of using the `backColor` property
  - A complete version of this revised movie is available as "changeTextColor2.dir" on the 337 server in the folder labeled "_Movies"

Create a movie with Push Buttons to Start/Stop sound file playback

Lingo allows many advanced capabilities for handling sound. However, at their most basic level, sounds can be incorporated very easily using the Score

- File→New→Movie
- Make certain that the "Span Duration" setting for new sprites is set to "1"
  - File→Preferences→Sprite
  - set the "Span Duration" setting to a value of "1"
    - this will ensure that when you add a sprite to the Stage or Score, the duration of the sprite is only a single frame
- Create two Push Buttons
  - Insert→Control→Push Button
  - type the label "Play"
  - Insert→Control→Push Button
  - type the label "Stop"
    - adding the Push Buttons in this way ensures that they are inserted at exactly the same location which will prove desirable for the purpose that these buttons are to serve
- Notice that the Play button was added to the first frame of Sprite channel number 1 in the Score and the Stop button was added to the first frame of Sprite channel number 2
- Click and drag the Stop button from frame 1 to frame 2 of Sprite channel number 2 in the Score
  - if you playback your movie now (with "Loop Playback" turned on) you will see a rapid alternation of the two buttons as the playback head alternates between frame 1 & 2
- Import a sound file
  - File→Import
  - Locate an appropriate file, then import it using the "Standard Import" option
- Click on the sound file cast member (number 3) in the Cast Window and drag it to frame 2 of Sound Channel 1 in the Score; Sound channel 2 would work just as well for this example, but just to be consistent put it in Sound channel 1
  - this means that whenever the playback head is in frame 2 the sound file will playback, but while the playback head is in frame 1, the sound file will not play
- Add brief Lingo script commands to the Push Button cast members
  - Play button

        - click on the Play Button cast member and provide the name "Play Button"
        - click on the Cast Member Script icon and type the following command into the **`on mouseUp`** event

```
on mouseUp
  go to frame 2
end
```

    - Stop button
        - click on the Stop Button cast member and provide the name "Stop Button"
        - click on the Cast Member Script icon and type the following command into the **`on mouseUp`** event

```
on mouseUp
  go to frame 1
end
```

- Finally, add one final Frame channel script
    - double-click on frame 1 of the Frame channel in the Score
    - type the following into the Script Window that appears

```
on exitFrame me
  go to the frame
end
```

    - notice that the newly created script you created appears as member number 4 in your Cast Window and is identified as a "behavior" (see the "picon" in the lower right-hand corner of the member slot?) … Congratulations, you just created your first behavior!!
- Now click and drag this behavior from the Cast Window to frame 2 of the Frame channel in the Score
    - you could have just double-clicked on this same frame and typed in another identical script, but that would result in adding an unnecessary cast member since you already have a cast member (the behavior in cast member number 4) that accomplishes what you want … reuse such elements to minimize the size of your resulting movie. It wouldn't really make a difference in a small movie like this one, but as you begin to add more and more elements to your movies, such duplication – if there is a significant amount of it – can negatively impact the performance of your movie.
- A complete version of this movie is available as "playSound.dir" on the 337 server in the folder labeled "_Movies"
    - **Challenge**: create markers to identify frame 1 and frame 2 then use the marker names in your Push Button scripts instead of the frame numbers
        - A complete version of this revised movie is available as "playSound2.dir" on the 337 server in the folder labeled "_Movies"

Behavior Inspector

Now that you have added your own behavior to a movie, this would be a good time to explore the capabilities of the Behavior Inspector further. If you double-click on the behavior cast member (cast member number 4 in "playSound.dir"), the Property

Inspector will pop up. If it is already open, it will receive focus. Before proceeding make certain that all options are available, by clicking on any right-facing triangles so that the Behavior Inspector expands, providing additional options

Take time to explore the options available in the three primary areas of the Behavior Inspector:

- Behavior Pop-up – add new behaviors or select already existing ones
- Event Pop-up – what event will trigger the behavior?
- Action Pop-up – what happens when the trigger event occurs?