Beyond Flash: Using Director's more complete sound handling capabilities

©Dr. Scott D. Lipscomb, Associate Professor
Northwestern University School of Music
lipscomb@northwestern.edu

Web Site Related to this Presentation:
http://faculty-web.at.northwestern.edu/music/lipscomb/nsmit/

Using Director's Built-in "Behaviors"

Basic Director Terminology

- stage
- cast
- score
- cursor
- sprite
- behaviors
- rollover
- Lingo

Playing a Sound (use "`PlaySoundTemplate.dir`")

- Go to the frame that contains the sprite you want to use as a trigger to play the sound file; it can be either an internal or external sound file, but for an external file, you must provide the complete path to the file[1]
- Select Window→Library Palette
- Add the "`Play Sound`" behavior to your movie
    - Open the "`Media`" submenu of the library, then the "`Sound`" submenu, and drag the behavior named "`Play Sound`" and drop on the sprite with which you wish to trigger the sound file
    - In the entry box, type the following information
        - the *exact name* of the sound file; if the sound file has been imported into the movie cast (the easiest—and recommended—way), simply select the appropriate file from the drop-down list labeled "`Sound to play`";
        - sound channel number (the default of "1" is usually fine)
        - select the action that you wish to trigger playback of the sound file from the drop-down list labeled "`When to play sound`"; in this case, it should be "when the mouse clicks on the sprite"

---

[1] There are problems with using external files about which you, as a creator, must be aware. First, there are cross-platform issues due to the use of "\" on the PC and ":" on the Mac in the path to the file. Second, the external file *must* exist on any computer upon which your movie is played … in the *exact* same location on the hard drive. Generally, it is much wiser to use internal sounds unless you are using extremely large sound files that will be distributed with your movie and/or you are an advanced user who clearly understands the processes involved.

- ▪ enter a number of times to play the sound; "1" is usually the best choice, but you can repeat ("loop") the sound as many times as you wish. To have the sound file loop indefinitely—usually a *very* bad idea—enter "0" into this box
- ▪ Now return Director's playback head to the beginning of the movie. When the frame containing your playback button appears, you will be able to play the sound file by clicking on the appropriate sprite

Rollovers (use "`MouseOverTemplate.dir`")

- ▪ Go to the frame that contains the sprite you want to use as a trigger to play the sound file
- ▪ Select Window→Library Palette
- ▪ Add the "Play Sound" behavior to your movie
  - o Follow instructions above to select the appropriate sound file, but in the "`When to play sound`" drop-down list select "`when the cursor moves over the sprite`"
- ▪ Add the "`Rollover Member Change`" behavior to your movie
  - o Open the "`Animation`" submenu of the library, then the "`Interactive`" submenu, and drag the behavior named "`Rollover Member Change`" and drop on the sprite with which you wish to trigger the sound file
  - o In the selection box that appears, choose the image that you would like to appear when the mouse "rolls over" the sprite[2]
- ▪ Add the "`Rollover Cursor Change`" behavior to your movie
  - o Open the "`Animation`" submenu of the library, then the "`Interactive`" submenu, and drag the behavior named "`Rollover Cursor Change`" and drop on the sprite with which you wish to trigger the sound file
  - o In the selection box that appears, choose the cursor that you would like to appear when the mouse "rolls over" the sprite

Fundamentals of Digital Sound

File Formats

- ▪ Director imports AIFF and WAV sounds (both compressed and uncompressed), AU, Shockwave Audio, MP3, and Macintosh System 7 sounds.
- ▪ Director can stream the following sound file formats:
  - o QuickTime, Shockwave Audio, and MP3 sounds that are linked via a URL
  - o QuickTime, Shockwave Audio, MP3, AIFF, WAV, and AU sounds that are linked to a local file
  - o Director can also use native Macintosh sounds, but since these are 8-bit sounds their quality is low and it is best to use another option

---

[2] Though there is an image called "Button-over.gif" in the template created for this presentation, when you create your own movies, you will need to create versions of each image for each interactive feature you wish to incorporate (e.g., mouseovers, mouse down, etc.). There are several programs that make this process extremely easy, e.g., Macromedia Fireworks, Adobe ImageReady, etc.

Sample Rate

- For best results and to ensure cross-platform compatibility, use sounds that have 16-bit depth and a sampling rate of 44.1, 22.050, or 11.025 kHz. The higher the sampling rate, the better the sound quality.

Sound Bit Depth

- For best results, use 16-bit sounds only
    - Any sounds that use Shockwave compression, will automatically be converted to 16-bit, whether the original source is 8- or 16-bit

Stereo vs. Mono

- Reducing a sound to mono immediately reduces the file size by half; this is a good idea unless you have special requirements for which the stereo image is necessary
    - High-quality (i.e., at least CD-quality) is required; music and/or sounds are the main focus of your movie, rather than just background sounds
    - Your sound files make use of stereo effects, e.g., panning

Calculating Audio File Size (in bytes)

- Mono File Size: LengthInSeconds * SampleRate * (BitDepth ÷ 8)
- Stereo Files: 2 * MonoFilesSize
- Online Calculator available at:
  http://faculty-web.at.northwestern.edu/music/lipscomb/imea2002/AudioFileSizeCalculator.htm

Audio Compression

As you will find when using the calculator referenced above, files can become very large quickly. In order to reduce the file size, you may wish to consider the possibility of using some form of compression. Just like with graphic formats like JPEG, there is always a trade-off between the resulting quality of the compressed file and its size. Small file sizes can equate to terrible sounding audio.

Another possibility you may wish to consider is streaming the audio to your listeners instead of having them wait for an entire file to download. For details on this process and to find a comparison of the various file formats and streaming media, you can access the resource created for my recent presentation at the Conference of the Association for Technology in Music Instruction at: http://faculty-web.at.northwestern.edu/music/lipscomb/atmi2001/

Adding a Background Sound (use "`BackgroundSoundTemplate.dir`")

- Import a sound file
    - Select an empty cast member in which to place the imported sound file
    - Select File→Import and find the file you wish to use
    - Highlight the file name and click on the "`Import`" button
- The Sound Properties window
    - Window→Inspectors→Property
    - Member tab
        - Member name
        - Comments

- Unload options
- Edit (in Director or External Sound Editor)
  - o Sound tab
    - Loop
    - Audition sound file (Play/Stop)
  - o Cast
    - Cast name
    - Preload options
- Drag the cast member onto Sound Channel number 2 at frame 1 of the Score[3]
  - o When adding a background sound, make sure that the length of the sound file in the stage window starts from frame 1 and extends to the end the movie; otherwise the playback results may be other than intended.

Triggering Sound

puppetSound command

When using this method, a sound channel is made a "puppet," allowing Lingo (Director's scripting language) to "pull the strings," overriding instructions in the Score. To play a sound, you use the following syntax: the command followed by two parameters (sound channel number and either the number or name of the sound cast member)

```
puppetSound 1, "Beep" -- where "Beep" is the name of a cast member
puppetSound 1, 4 -- where 4 is the number of the sound cast member
```

To cause a sound to stop playing, send the `puppetSound` command with the sound channel number and "0" as the second parameter

```
puppetSound 1, 0
```

Adding puppetSound to Your Movie (Use "`puppetSoundTemplate.dir`")

- Go to the frame that contains the sprite you want to use as a trigger to play the sound file (this can be any frame of the movie template provided)
- Click on the sprite that will trigger the start of the sound file (the button on the left of the template)
- While the sprite has the `focus` (there will be a bounding box visible), open the "`Behavior Inspector`" … `Window→Inspectors→Behavior`
- Open the "`Behavior Popup`" menu by clicking on the "`+`" sign in the upper left-hand corner of the Behavior Inspector
- Click on "`New Behavior…`" and type "`Play`" into the text entry box that appears
  - o notice that this new behavior has now been added to the list of behaviors for the selected sprite
- Open the "`Event Popup`" menu by clicking on the "`+`" sign below and to the left of the newly added behavior and select "`mouseUp`" so the sound file will begin playing when the user releases the mouse button[4]

---

[3] It is important that you use sound channel number 2, since the button sounds we created previously are in sound channel number 1. If you also place the background sound in sound channel 1, then it will be interrupted when any of the mouse sounds occur

[4] This is generally a better idea than using the mouseDown event, since it allows the user to change her/his mind after clicking the mouse by moving the mouse away from the sprite before releasing the mouse button.

- Open the "**Action Popup**" menu by clicking on the "**+**" sign to the right of the "**Event Popup**" menu and select **Sound→Play Cast Member**
    - From the dropdown entry box that appears, select the appropriate sound file (if using the template, there will be only one choice … "Bach"), then click on "**OK**"
- A Behavior Script entitled "Play" has been added to your Cast and contains the following script (or something very close to it)[5]

    ```
    on mouseUp me
      puppetSound 3, member "Bach"
    end
    ```

    - This script causes the cast member named "**Bach**" to play on sound channel number 3

Now, let's add the behavior to stop playback of the sound file (less details will be provided for this process … if necessary, refer to the more detailed instructions above)

- Click on the sprite labeled "**Stop Sound**"
    - The Behavior Inspector should still be open; if not, follow directions above
- From the "**Behavior Popup**" menu, select "**New Behavior**…" and type "**Stop**" into the text entry box
- From the "**Event Popup**" menu, select "**mouseUp**"
- From the "Action Popup" menu, select Sound→Play Cast Member
    - I know, I know … you're wondering "Why should I select "Play Cast Member," when what I *really* want to do is stop the sound file from playing?" We'll take care of it in the next step …
    - Double-click on the cast member named "Stop" so the Behavior script opens.
        - Delete the text **member "Bach"** and replace it with with a 0 (that's a **zero**, not the letter "O")

        ```
        on mouseUp me
          puppetSound 3, 0
        end
        ```

        - Setting the cast member to "0," as you just did, causes the sound playing back in the referenced channel (number 3 in this example) to stop playing

You will notice as you play around with this movie that every time you click on the "Start Sound" button, Bach's *Toccata and Fugue in D minor* starts playing at the beginning … in fact, it might be just repetitive enough to drive you absolutely bats within a couple of minutes. What if, instead, you want the sound file to pause when the "**Stop**" button is pressed and resume playback when the "**Play** "button is pressed? You're undoubtedly thinking … "There's got to be an easy way" … and you're *almost* right. It's not exactly easy, but it's not that hard either … so let's do it …

- Using the file you created in the previous section as a starting point, we need to make a couple of minor adjustments, requiring that you actually type some **code** into a script. Don't be intimidated … just roll up your sleeves and take a deep breath. Remember, you learned to play a musical instrument … you can do *anything*!!! [It may help to keep saying that over and over throughout the next section.]
- First, double-click on the cast member named "**Play**". When the script window opens, edit the text, so that it reads exactly as follows:

---

[5] To view the contents of the Behavior Script, simply double-click on the cast member named "Play"

```
on mouseUp me
   sound(3).play()
end
```

- o There, you have now used *Lingo* (Director's scripting language) to access the "sound" object and utilize its "play" function
    - ▪ if you understood that last sentence it may be time to consider changing occupations to the much higher-paying field of computer programming.
- o If you're like most of us and the bullet point above seemed about as coherent as "Τηερε, ψου ηαϖε νοω υσεδ *Λινγο* (Διρεχτορσ σχριπτινγ λανγυαγε) το αχχ εσσ τηε σουνδ οβφεχτ ανδ υτιλιζε ιτσ πλαψ φυνχτιον" (yup, that's Greek), then let me put it in more understandable terms
    - ▪ Remember that the sound file is playing in sound channel number 3 … that's the meaning of the "3" in our original **puppetSound** script.
    - ▪ The "**sound(3)**" part of the script you just typed lets Director know that you are intending to do something with the sound it associates with that channel; in this context, the "3" is called a *parameter* (or *argument*)
    - ▪ The period (".") simply separates the "thing being controlled" (the sound object) from the "intended action" (initiating playback)
    - ▪ the "**play()**" part of the script tells Director what action you expect to take place
        - • The ending parentheses ["()"] are necessary to let Director know that this is a *function* (the programmer's term for an action)
- o These are the same basic fundamentals (objects, events, and parameters) used by C-programmers to create word processors and notation software … are you feeling smart now?!![6]
- ▪ To continue, simply double-click on the cast member named "**stop**" and replace the script with the following text:

```
on mouseUp me
   sound(3).pause()
end
```

- o Using the explanation provided above, can you figure out the meaning of each component of this script? … it's easy once you know the basics, isn't it?
- o Notice that you use the "pause" function instead of the "stop" function for this purpose. The stop function is a more drastic measure and is not needed in this context.
- ▪ Finally, click on the first vacant cast member slot and open the Script window:
    Window→Script
- ▪ Type the following text … *exactly*:

```
on startMovie
   puppetSound 3, "Bach"
   sound(3).pause()
end
```

- o Before closing the script window, open the Property Inspector (**Window→Inspectors→Property**) to ensure that you have created a *Movie* script

---

[6] You should be aware that there is a "sissy" way to write many scripts using what Lingo refers to as its "verbose" form. For example, instead of writing **sound(3).volume = 150**, to change the loudness level of sound channel 3, you could write **set the volume of sound 3 to 150**. However, as you begin to utilize more advanced capabilities, there is rarely a verbose alternative. Learning the "dot syntax" described above will pay off.

- In the Property Inspector window, click on the "Script" tab and make sure that the drop-down box labeled "Type" is set to "`Movie`"
  - Now, play the movie and it should allow you to pause & resume as you wish
    - o If you are really industrious, you could do a couple of things to clean up the state of our movie:
      - change the button labels to "Resume Sound" and "Pause Sound," respectively
      - add a button that will "Reset" the sound file to the beginning … I'll let you figure that one out on your own

Types of Scripts (or "Why Do I Need to Know This Stuff?")
Sometimes the type of script you create can mean the difference between a working movie and one that almost works! For example, if the "`startMovie`" script you created above were a Behavior script instead of a Movie script, the sound would not have played … though the buttons would happily perform their `mouseOver` and `mouseDown` image functions. [Go ahead, try it … I'll wait.] As a result, it *is* a fairly important topic to understand. Outlined below, are the basic differences between the script types (in Director 8.5). For the sake of clarity—I tend toward verbosity, if you haven't noticed—I have extracted these definitions verbatim from the Director documentation and provide them here for your convenience:

****

Director uses four types of scripts: behaviors, movie scripts, parent scripts, and scripts attached to cast members. Behaviors, movie scripts, and parent scripts all appear as cast members in the Cast window.

Behaviors—are scripts that are attached to sprites or frames in the Score, and are referred to as sprite behaviors or frame behaviors. The Cast window thumbnail for each behavior contains a behavior icon (" ") in the lower right corner. When used in [the Director documentation], the term "behavior" refers to any Lingo script that you attach to a sprite or a frame. This differs from the behaviors that come in Director's Library Palette. All behaviors that have been added to the cast appear in the Behavior Inspector's Behavior pop-up menu. (Other types of scripts don't appear there.) You can attach the same behavior to more than one location in the Score. When you edit a behavior, the edited version is applied everywhere the behavior is attached in the Score.

Movie scripts—respond to events such as key presses and mouse clicks, and can control what happens when a movie starts, stops, or pauses. Handlers in a movie script can be called from other scripts in the movie as the movie plays. A movie script icon (" ") appears in the lower right corner of the movie script's Cast window thumbnail. Movie scripts are available to the entire movie, regardless of which frame the movie is in or which sprites the user is interacting with. When a movie plays in a window or as a linked movie, a movie script is available only to its own movie.

Parent scripts—special scripts that contain Lingo used to create child objects. You can use parent scripts to generate script objects that behave and respond similarly yet can still operate

independently of each other. A parent script icon ("⬚") appears in the lower right corner of the Cast window thumbnail.

Scripts attached to cast members are attached directly to a cast member, independent of the Score. Whenever the cast member is assigned to a sprite, the cast member's script is available. Unlike behaviors, movie scripts, and parent scripts, cast member scripts don't appear in the Cast window. However, if Show Cast Member Script Icons is selected in the Cast Window

Preferences dialog box, cast members that have a script attached display a small script icon ("⬚") in the lower left corner of their thumbnails in the Cast window.
****

soundBusy command (use "`soundBusyTemplate.dir`")

At times, you may want elements (sprites) of your movie to behave differently when a sound is playing than they do when no sound is present. Lingo provides a way to determine if a specific sound channel is currently occupied with a sound file (in addition to actually playing the file, it could be temporarily paused, i.e., still "busy"). The `soundBusy` command allows you to easily test for this property.

- Open the "soundBusyTemplate.dir" file
- Double-click on Frame 1 of the Score's Script channel
    - In the empty script window that appears, type the following text:

```
on exitFrame me
  if soundBusy(3) then
    --Play button
    sprite(1).visible = FALSE
    --Play label
    sprite(3).visible = FALSE
    --Stop button
    sprite(2).visible = TRUE
    --Stop label
    sprite(4).visible = TRUE
  else
    --Play button
    sprite(1).visible = TRUE
    --Play label
    sprite(3).visible = TRUE
    --Stop button
    sprite(2).visible = FALSE
    --Stop label
    sprite(4).visible = FALSE
  end if
end
```

- Notice that, each time the score's playback head exits frame 1, this `If … then` statement tests whether the sound file is currently "in action" by using the `soundBusy` command. If the sound channel is currently in use [i.e., `soundBusy(3) = TRUE`], then the `visible` property of both the Play button and the Play button label are set to `FALSE`, while the `visible` property of both the Stop button and the Stop button label are set to `TRUE`. If the sound channel is not currently in use (i.e., the sound file has been stopped, the `visible` property of

both the Play button and the Play button label are set to **TRUE**, while the **visible** property of both the Stop button and the Stop button label are set to **FALSE**.

This command is also useful if you need to have more than one sound playing simultaneously. Director allows up to eight sounds to be playing simultaneously. However, if a sound is playing in a given channel and another sound is played using the same channel, the first sound file is interrupted. The first sound file will not begin playing again, even after the second file stops. As a result, it may be wise to create a routine that provides you with the capability to perform a search for a channel that is not busy (see "**soundBusyFindChannel.dir**"):

```
on getChannel
  repeat with channel = 8 down to 0
    if soundBusy(channel) then
      --do nothing
    else
      exit repeat
    end if
  end repeat

  --You, of course, will want to do something more useful
  --  with this information.
  if channel = 0 then
    alert "All channels are busy ... please, try back later."
  else
    alert "You may use channel " & channel & "."
  end if
end
```

soundDevice and soundDeviceList commands

In most cases, you can leave the **soundDevice** setting at its default setting. On Macintosh computers, in fact, there is only a single soundDevice … the MacSoundManager (the only one you need). Windows computers might have as many as three devices available. In order of preference due to the quality of reproduction, these are DirectSound, QT3Mix, and MacroMix. If you wish to see which devices are available on your computer, simply type the following into the **Message window**:

```
put the soundDeviceList
```

Adjusting the Loudness of Your Sound Files

Director allows you to change the volume during playback in two ways: a global adjustment that alters the sound level of all sounds on your computer (**the soundLevel**) or setting the level for a specific sound channel in Director (**the volume**). You can also use **the soundEnabled** property to turn all sounds on or off.

the soundLevel

If your movie makes a global adjustment to the sound level, it is wise to save the current level when your movie starts, so you can return the system to the same level when your movie completes

```
global currentLevel

on startMovie
  --save the current system sound level
  currentLevel = the soundLevel

  --set the level for your movie (from 1 to 7)
  the soundLevel = 5
end

on stopMovie
  --return the system to its previous level
  the soundLevel = currentLevel
end
```

the volume

The volume level for a sound channel can be set to any value between 0 and 255. You can use either the verbose or dot-syntax version:

```
the volume of sound 3 = 225
```

or

```
sound(3).volume = 225
```

As you develop cross-platform applications you should be aware that sound levels are nowhere near consistent across—or even within—platform.

fadeIn and fadeOut commands (use "`FadeInFadeOutTemplate.dir`")

These two commands allow you to effectively blend sounds by fading them in and out. You simply identify the channel you wish to operate upon and the amount of time (in milliseconds[7]) that you want the fadeIn or fadeOut to take. Let's add this capability to our Start & Stop button movie.

- ▪ Open the "`FadeInFadeOutTemplate.dir`" movie
  - o You will notice that two buttons—appropriately labeled "Fade In" and "Fade Out" have been added
- ▪ Click on the "Fade In" button and add a new Behavior using the Behavior Inspector
  - o From the "`Behavior Popup`" menu, select "New Behavior" and type "fadeInSound" into the text entry box
  - o From the "`Events Popup`" menu, select "`Mouse Up`"
  - o Double-click on the cast member labeled "fadeInSound" to open the Script window and add the following text to the "on mouseUp" event
    ```
    sound(3).fadeIn(5000)
    ```
    - ▪ this line of Lingo script tells Director that you want the sound playing in sound channel 3 to fade in over a period of 5000 milliseconds (5 seconds)[8]
- ▪ On your own, add a "fadeOutSound" Behavior to the "Fade Out" button
- ▪ Now, play the movie and click on the "Start Sound" button. Notice how pressing the appropriate button causes the sound to fade in or out

---

[7] Previous versions of Director used "ticks" (1/60th of a second) instead of milliseconds.
[8] okay, okay … the verbose version would be `sound fadeIn 3, 300`. Notice that, instead of milliseconds, when using this version you must calculate the number of "ticks" (1/60th of a second).

Synchronizing Sound (use "`SonataFormTemplate.dir`")
One of the most impressive ways that the sound capabilities of Director can be utilized in an educational context is the synchronization of the audio and visual components. In order to take advantage of these capabilities, you must use an external sound editor (Sound Forge or Cool Edit on the Windows platform or SoundEdit 16 or Peak on the Macintosh) to add "markers" to the sound file. Taking full advantage of these markers—or "cuePoints," as they are called in Director—will be much easier if you assign meaningful names. [Demo Sound Forge]

Once you have added markers to your sound file and saved it (preferably in either AIFF or WAV format), import the sound file into your Director movie. You can now use some powerful techniques to access your sound file, navigate to a specific location, or cause events to occur based on the position of the playback head within the sound file. The primary tools provided for this purpose are:

on cuePassed—this message is sent each time a cuePoint is passed. The parameters passed by the message include the channel number, the cue number (i.e., the nth marker contained in the sound file), and the current marker name. The "`SonataFormTemplate.dir`" movie uses this message to great advantage, as you will see below.

mostRecentCuePoint—this property gives you access to the name of the cuePoint passed most recently, using the following syntax to save this information in a variable named "marker":
```
marker = sound(3).mostRecentCuePoint
```

isPastCuePoint—this function can be used in two ways, using an integer or name to represent the cuePoint:
```
--using an integer results in a value of TRUE if the
--  cuePoint has been passed or FALSE if not
sound(3).isPastCuePoint(1)

--using a cuePoint name, the function returns the number
--  of times the cuePoint has been passed
sound(3).isPastCuePoint("Introduction")
```

In the excerpted script below, each text item representing a section of the sonata form movement on the stage is set to black italic font, except for the text referencing the current marker position which is set to red bold-faced type:
```
on cuePassed channel, cueNumber, cueName
  --set all headings to normal text
  --  cast members 7-21 are text labels for
  --  each section of the sonata form movement
  repeat with counter = 7 to 21
    if member(counter).color = rgb(170,170,170) then
      --do nothing
    else
      member(counter).fontStyle = [#italic]
      member(counter).color = rgb(0,0,0)
    end if
  end repeat

  --set color of current section to red, bold-face font
  if cueName <> "End" then
```

```
                    member(cueName).fontStyle = [#bold]
                    member(cuename).color = rgb(255,0,0)
                  end if
                end
```

The `cuePointNames` and `cuePointTimes` properties provide easy access to a list of all of the marker names and times contained in a sound file. These can easily be stored in a list for later access, as shown below for a sound in channel 1:

```
            gListMarkerNames = member(1).cuePointNames
            gListMarkerTimes = member(1).cuePointTimes
```

When any text item representing a section of the sonata form movement on the stage receives a mouse click, the following script simply cycles through the list of marker names contained in the global list of marker names (`gListMarkerNames`) until it finds the one that matches the `MarkerName` argument. This argument is simply the cast member name associated with the sprite that was clicked … a result of careful planning. Once a match is found, the "counter" value is retained by "exiting" the repeat structure and that same value is used to access the appropriate time value from the global list of marker times (`gListMarkerTimes`). This time value is then used as the `#StartTime` parameter of the sound object's `play()` function.[9]

```
            global gListMarkerNames, gListMarkerTimes


            on setPlaybackPos MarkerName
              repeat with counter = 1 to gListMarkerNames.count
                tmp = getAt(gListMarkerNames, counter)
                if tmp = MarkerName then exit repeat
              end repeat

              MarkerTime = getAt(gListMarkerTimes, counter)

              sound(1).play([#member: member(1), #StartTime: MarkerTime])

            end
```

Using "`SonataFormTemplate.dir`"

Templates are a valuable time saver. If you ever create something that you intend to use over & over, consider creating a template that allows you to create alternate versions in a matter of minutes. One example is provided below. The sonata form template is available to you online at: http://faculty-web.at.northwestern.edu/music/lipscomb/imea2002/.

Follow these simple steps:
- Using sound editing software (Sound Forge, Cool Edit, SoundEdit 16, or Peak), add the following markers to a sonata form sound file;

```
            Introduction
            ExpPrimaryTheme
            ExpTransition
            ExpSecondThemeGroup
```

---

[9] The `play()` function used previously has many optional parameters, providing a *great* deal of control of sound playback, including StartTime and EndTime. The complete list of parameters is:
`sound(channelNum).play([#member: member(whichmember), {#startTime: milliseconds, #endTime: milliseconds, #loopStartTime: milliseconds, #loopEndTime: milliseconds, #loopCount: numberOfLoops, #preloadTime: milliseconds, #rateshift: shiftAmount}])`

```
ExpClosingTheme
ExpPrimaryTheme2
ExpTransition2
ExpSecondThemeGroup2
ExpClosingTheme2
Development
RecapPrimaryTheme
RecapTransition
RecapSecondThemeGroup
RecapClosingTheme
Coda
End
```

- o These marker names must be entered *exactly* as listed above
  - Any marker name that is not used will result in a "grayed-out" text label for the associated sonata form section in the resulting movie
- o Once all marker names have been added, save the sound file using an appropriate name
- Start Director
- Open "`SonataFormTemplate.dir`"
- Replace the sound file in cast member 1 ("`TestSoundFile`") with the sound file you just saved
  - o Delete cast member 1 by clicking on it once then pressing the `DELETE` or `BACKSPACE` key
  - o From the "`File`" menu, select "`Import…`" and navigate to the intended sound file; select the sound file and click on the "`Import`" button
- Play your new movie … that's all there is to it! It's as easy as 1, 2, 3 (4, 5) …

Creating a Shockwave Version of Your Movie
It takes only a single mouse click and a filename!!
- Click on the "Publish" icon in the toolbar (or select `File→Publish`)
- Provide a filename for your Shockwave movie
  - o it's best of this filename (like documents intended for the web) contains no spaces

Contact Information:
Dr. Scott D. Lipscomb
Northwestern University
School of Music, MAB 119
711 Elgin Road
Evanston, IL 60208
lipscomb@northwestern.edu
http://faculty-web.at.northwestern.edu/music/lipscomb/

Useful References

Books

Gross, Phil (1999). <u>Macromedia Director 7 and Ligo Authorized</u>. Berkeley, CA: Macromedia Press.

Roberts, Jason & Gross, Phil (1999). <u>Director 7 Demystified: The Official Guide to Macromedia Director, Lingo, and Shockwave</u>. Berkeley, CA: Macromedia Press.

Steinhauer, Lauren (1997). <u>Macromedia Director 6 for Dummies</u>. Chicago, IL: IDG Books Worldwide

Web Sites & Listservs

Chinwag Discussion List for Developers Using Shockwave and Director Tools
http://www.Chinwag.com/html/mailing_lists_o.html

Direct-L Listserv
To subscribe, send an email message consisting of solely the following two lines (no subject and no signature file) to: listserv@uafsysb.uark.edu
SUBSCRIBE DIRECT-L [your name and email address]
SET DIRECT-L DIGEST

Director Demystified Web Site
http://www.demystified.com

Director Online
http://www.xtramedia.com/lingoTips.shtml

Director Technotes
http://www.macromedia.com/support/director/ts/nav/
http://www.macromedia.com/support/director/dev_index.fhtml

Director Web
http://www.mcli.dist.maricopa.edu/director/

Lingo Help: DaLingoKid
http://www.jervo.com/pages/dalingokid.html

Lingo-L Listserv
http://www.penworks.com/LUJ/lingo-l.cgi

Lingo Users Journal
http://www.penworks.com/LUJ/index.html

ShockRave
http://www.Shockrave.com

UpdateStage
http://www.updatestage.com