Using sound in Flash to enhance the music learning experience: A demonstration and workshop

Scott D. Lipscomb, Associate Professor
School of Music, *Northwestern University*

One of the great challenges facing educators at present is the task of making the classroom more engaging for students. How can we most effectively get them actively involved in the learning process, rather than sitting passively as the instructor lectures on the topic of the day or rehearses that same piece again and again. It is possible that potential solutions to this dilemma might be found in a place considered most unlikely by many educators: video games. I can almost sense the uncomfortable stir when this topic is mentioned, as many begin too regurgitate oft-heard statements such as "Video games are just a waste of time" or "Most games are full of violence." To initiate this discussion, however, I am going to ask that we refrain from such expository statements and focus on what it is, specifically, that young people find attractive about these games, many of which are highly complex in their nested structures, requiring high-level thinking skills to navigate their virtual domains. To put it concisely: "What rewards and satisfaction do video games offer our students that we are failing to provide them in the classroom?"

In order to be successful at playing a game (i.e., to "win"), one must be able to learn and adapt to the rule system created by the designer and apply the necessary problem-solving skills to safely navigate through the virtual "world" of the game. This problem solving process often requires impressively advanced critical thinking skills, as various options are considered. This is true whether the goal is to arrange water pipes in the world of *Myst* in order to allow access to a remote location or to determine the best method of hijacking a car in *Grand Theft Auto*. Though we can certainly agree that one goal is more admirable and socially acceptable than the other, similar critical thinking and problem-solving skills are involved in each instance.

I would like to propose that we, as instructors, take seriously the charge to make our instructional materials more engaging for our students. Multimedia instructional materials afford one possible means toward meeting this goal. When developing such materials, user interaction can be a designed requirement … unless the user interacts with the interface, nothing happens. Rather than passively watching a slide show, for example, at least a modicum of engagement is forced upon the student. Though we do not yet have definitive proof that technology enhances learning, research suggests that there are few cases in which the presence of technology, when effectively incorporated, has proven detrimental to learning. Likewise, this same research continues to reveal that student attitudes toward the use of technology are consistently positive (Webster, 2002). As every teacher knows, a positive attitude provides a large step in the right direction.

There are, of course, varying levels to which interactivity can play a role in the learning experience. It is possible to design a set of materials that require the user to click on a button (or provide some other input) before proceeding from one section of the materials to another. This low level of interactivity leaves the user sitting passively between these sequential interactions. Another possibility, in which a higher level of activity is required, would be to require the user to click to initiate an action, click again to stop the action or initiate another action. The much higher level of interaction evident

-

in video games, however, allows the user to interact in a manner that has consequences for future actions, resulting in a series of decisions that ultimately leads either to success or failure. Fortunately, in the context of video games, failure means simply that the user must start the game over or return to a recently saved location within the game … quite different from the result of failure in a real life context.

During the past decade, I have integrated technology and student interactivity more and more into my music classroom. I began by simply providing streaming audio and video, allowing students to access sound files 24 hours a day, seven days a week. This kind of start-stop interaction provided a high level of convenience, but lacks the kind of engagement considered necessary for deep learning. For demonstration purposes, using *Director* and *Flash*, I then created numerous animations to illustrate complex concepts like musical form and the Gestalt principles as they relate both to visual and auditory cognition. Most recently, I have begun creating multimedia materials that allow my students the opportunity to navigate from one point to another within a piece of music and to explore the composition in a manner that would not otherwise be possible, rather than enforcing upon the student a linear (i.e., beginning-to-end) listening experience. Some of these same tools can be provided to advanced students to enable them to use there higher level skill set to create their own projects and demonstrations.

A Tool of Choice: Macromedia's *Flash*

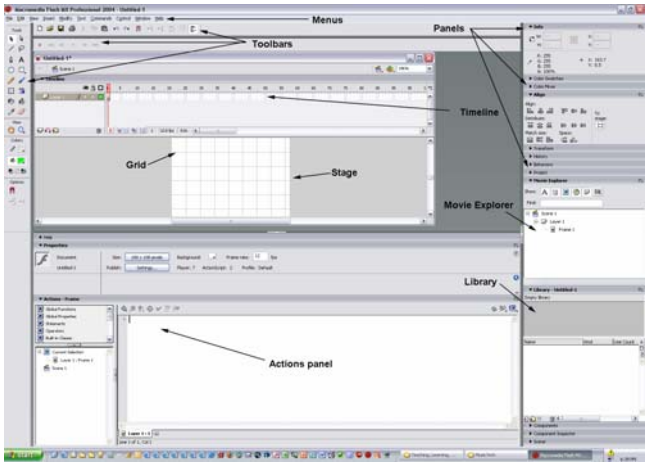
Macromedia *Flash MX 2004* provides an authoring environment for the creation of truly interactive multimedia materials.[1] The free browser plug-in and player available from the Macromedia web site have established a near-ubiquitous presence on the Internet … over 97% of the computers used to surf the WWW have the plug-in installed, making this program one of the most reliable ways to ensure that your users will be able to utilize your interactive instructional materials to their fullest extent. *Flash* provides a marvelous balance between user-friendliness and complexity. It is possible to learn the program's most basic capabilities and put these to work almost immediately in the creation of interactive instructional materials. For those who are willing to spend the extra time to learn ActionScript, the scripting language that is part of *Flash*, the program's potential is almost limitless. One disappointing limitation is that *Flash* does not include MIDI capability, though it masterfully handles numerous types of digital audio files and substantially reduces file size of the web-ready movies using effective compression algorithms.

Though it does not work with MIDI, *Flash* offers many other attractive and useful capabilities. *Flash* allows users to import graphic images, video, sound files, and many other types of media created in other programs or downloaded from the Internet. Many types of objects (e.g., text, ovals, rectangles, lines, customized gradients, etc.) can be created directly in *Flash*, using the Tool palette. These basic shapes and objects can be combined to create professional looking graphics for use in your own animations. In addition, *Flash* comes with several "Common Libraries" to provide many essential tools. The "Buttons" library, for example, contains an impressive array of buttons, sliders, knobs, faders, and other useful objects that can be readily included in your movies. The "Learning Interactions" library contains templates – including complex interactive

[1] The opening paragraphs of "The Workshop" section of this paper are excerpted, in abridged form, from a forthcoming chapter (Lipscomb & Walls, in press) to be published in the *TI:ME Technology Guide.*

capabilities – for creating various types of quiz forms: true-false, multiple choice, fill in the blank, drag-and-drop, and others. In addition, the program comes with an impressive set of components with built-in functionality. Some of the most commonly-used Components – Macromedia's built-in, interactive objects – include the checkbox, radio button, progress bar, listbox, combo box, text input, and many others.

Like many Macromedia programs, *Flash* uses a "Timeline" metaphor to facilitate the creation of animations. This allows complex animations to be created without necessitating the frame-by-frame creation of each individual image. Instead, the user marks important locations (called "keyframes") along the Timeline and sets the desired location of each object in the movie at this point in time. *Flash* then automatically calculates the appropriate location for every object for every intervening frame, a process known as "tweening." Because *Flash* utilizes vector graphics instead of bitmaps (or raster graphics), the resulting files are much smaller and take less time for online visitors to download. The context-sensitive Properties window allows users direct access to the most common attributes of any object selected in the Work Area. Other panels (accessible from the "Windows" menu) allow the user to set alignment, transform & skew objects, mix colors, use pre-made Components, view all objects in the present movie, along with many other possibilities. For those who wish to move to more advanced levels of interactivity, *Flash* provides a powerful programming language called *ActionScript*, providing many of the same capabilities that were previously only the purview of much more complex and hard to learn programs like Java and C++.



There are two primary file types with which you must be familiar when working with Flash. When creating a Flash file (called a "movie") using the Flash interface, you work with an ".fla" file … the native format for Flash movies. When you "publish" a web-ready version of the movie, however, it is saved in ".swf" format (pronounced "swiff"). It is the ".swf" file that must be uploaded to your web site, *not* the ".fla" file. An internet browser (Internet Explorer, Netscape, Mozilla, Safari, Foxfire, etc.) with the Flash plug-in installed[2] will readily play back ".swf" file. In all likelihood, the browser will not know what to do if you link to an ".fla" file.

Work Flow in Flash

As you work with Flash, you will find that there are typically a series of steps involved with creating interactive multimedia content. The most common sequence of procedures is outlined below:

[2] You can download the free Flash plug-in from: http://www.macromedia.com/shockwave/download/download.cgi?P1_Prod_Version=ShockwaveFlash.

- draw or import graphics & sound files into Flash
- transform buttons, independent animations, & any other elements you intend to reuse into "symbols"
- place movie elements (vector graphics, bitmaps, symbols, etc.) on stage
- attach actions to buttons, movie clips, or frames on the timeline to make them interactive
    - in the steps outlined below, you will find that you are strongly encouraged to create two specialized layers in your Flash movies: an "action" layer (to hold ActionScript elements) and a "sound" layer (providing a single location for your audio components)
- select a frame, symbol, stroke, fill, or text block on the stage to adjust its properties via different panels

The Workshop

Animation Basics

Creating Keyframes

- **`Insert→Timeline→Blank Keyframe`** – create brand new content in a new frame
- **`Insert→Timeline→Keyframe`** – duplicates contents of previous keyframe

To add a Motion Tween animation, follow these steps:

- in the layer containing the object(s) you wish to animate, simply click on the frame where you want your animation to begin
- when you create a new movie, the first frame of Layer 1 automatically has a keyframe; otherwise, you can add one (**`Insert→Timeline→Keyframe`**)
    - if your objects are not yet on the stage, *ensure that a keyframe on the appropriate layer of your movie is selected*, then drag the object(s) from the Library onto the Stage at the location you would like for them to appear
- click on the final frame where you would like the animation to end, then select **`Insert→Timeline→Frame`**
- right-click (Win) or Control-click (Mac) the first keyframe of this animated sequence to open the pop-up menu and select "Create Motion Tween"
    - at this point a dashed line will connect the first frame of your animation to the last in the Timeline … this lets you know that the process is not yet complete
- click once again on the final frame of your animation and move the object(s) to the location where you would like it/them to be at the conclusion of this animated sequence
    - now the dashed line should have changed to an arrow pointing from the first frame of your Motion Tween animation to the last

-

Sound in Flash

Importing Sound & Using It in Your Movie

- **`File→Open`**, then browse to & select the **"01soundCrash_template.fla"** file[3]
- in the "sound" layer of the Timeline, click on the placeholder frame where you want your sound file to play (in this case, frame 20), then select **`Insert→Timeline→Keyframe`** (or F6)
    - you must *always* add a keyframe to the frames in your timeline where you want a specific event – like initiating playback of a sound file – to occur
- with the frame containing the newly added keyframe selected on the "sound" layer of the Timeline, *drag the sound file* ("CRASH.WAV") *from the Library panel onto the Stage*
    - notice that a representation of the sound wave appears in the frame to confirm that a sound has been added
- while the frame with the sound file is selected, notice the Properties window now contains additional options for manipulating the sound
    - from the "Effect" drop-down box, select Custom and notice how you can control the playback of your sound file; to get to this same window, you can simply click on the "Edit" button to the right of the "Effect" drop-down
    - set the "Sync" mode to "Event" and repeat "0" times
        - learn about the different "Sync" types (Event, Start, Stop, & Stream) to ensure that you get the most out of using sound in your Flash movies
- **`Control→Test Movie`**

Important capabilities to know:

- import a sound file to your library
    - **`File→Import→Import to Library`**, then browse on your hard drive to select a sound file
- view the content of your current movie's Library
    - **`Window→Library`** (or **`CTL+L`**)
        - notice that, after importing a sound file, it is automatically added to your movie's Library
        - if you click on the "Information" icon at the bottom of the Library window, you can determine the appropriate level & type of compression for your sound file

Adding Sound to a Button

- **`File→Open`**, then browse to & select the **"02soundButtons_template.fla"** file

[3] All templates used for this presentation are available from the author's web site. Simply point your Internet browser to: http://www.lecafeamericain.net/faculty.htm. From Dr. Lipscomb's home page, click on the "stuff" button, then click on the link to materials related to NSMIT 2005.

-

- double-click on the "Play" button to enter "Symbol Edit" mode; alternatively, you can select **Edit→Edit Symbols**
    - just above the very lefthand side of the Timeline, you should now see the name of the button upon which you are operating to the right of the "Scene 1" label
    - note that the Timeline for button symbols is different than the movie timeline, containing only four frames … one representing each button state
        - when you create your own buttons, in addition to the obvious button states, you must also make sure to designate some area of the button as the "hit" area (i.e., the area that responds to user interaction)
- decide which states (up, down, and/or over) to which you wish to attach sounds; for the current example do the following:
    - add a layer (**Insert→Timeline→Layer**) and label it "sound"
    - if your movie's Library window (*different from* the Common Library window) is not open, **Window→Library** (or **CTL+L**)
    - in the "sound" layer, click on the keyframe representing the state to which you wish to add a sound (in this case, the "Down" state), then drag the appropriate sound from the Library to the Stage
        - when creating your own buttons, you would repeat this step for every state within which you wish to add a sound
    - to exit "Symbol Edit" mode, click on "Scene 1" above the upper left corner of the Timeline panel
- To create interactivity that will allow a user to halt sound playback …
    - single-click on the "Stop" button instance
    - while the button is selected, add the following code to the "Actions" window:

```
on (release) {
      stopAllSounds();
}
```

- **Control→Test Movie**

Important capabilities to know:

- add a button from the Common Library to your movie
    - open the "Buttons" Common Library panel (**Window→Other Panels→Common Libraries→Buttons**)
    - select the button you want to use from the Common Library and drag an instance onto the Stage; the button is automatically added to your movie's Library

Control a Sound Using Frame Navigation

This method takes advantage of the "Start" and "Stop" sound Sync modes, initiating and terminating playback simply by navigating to an appropriate frame in the movie using the **gotoAndStop()** command when one of the buttons is pressed.

-

- **`File→Open`**, then browse to & select the **"03playSound_frameNavigation_template.fla"** file
    - note that a sound file ("Echo Flute 9.5.wav") has been imported into the movie's Library and the following layers exist on the movie's Timeline:
        - labels – used below to navigate using ActionScript commands
            - two labels have been provided for you:
                - "stop" in frame 1
                - "play" in frame 15[4]
        - actions – where frame scripts can be typed
        - main – the name given to the layer containing the buttons
        - sound – where the sound file will be placed
- click in frame 1 of the "actions" layer and notice that "**`stop();`**" has been typed into the Actions window (**`Window→Development Panels→Actions`**)
    - the purpose of this script is to avoid automatic playback of your movie immediately when it is loaded, turning complete control over to user interactions, as determined by ActionScript
- in the "sound" layer, add a keyframe (**`Insert→Timeline→Keyframe`**) at frame 15
- click on the keyframe in frame 1 of the "sound" layer and drag the sound file from your movie's Library to the stage, then click on the same keyframe a second time and select "Stop" from the "Sync" drop-down box in the Properties window
    - entering a frame containing a keyframe set to the "Stop" Sync mode causes the sound file, if playing, to halt playback
- now, click on the keyframe in frame 15 of the "sound" layer and drag the same sound file you dragged to frame 1 from your movie's Library to the stage, then click on the keyframe in frame 15 a second time and select "Start" from the "Sync" drop-down box in the Properties window
    - entering a frame containing a keyframe set to the "Start" Sync mode causes the sound file, if not already playing, to initiate playback
- the final step is to attach some basic ActionScript to the "release" event for each of the two buttons, causing the playback head of your movie to relocate to the appropriate frame in your movie, as determined by the frame label identified in the **`gotoAndStop()`** command
    - click on the "Play" button and type the following text into the "Actions" window:

```
on (release)
{
   gotoAndStop("play");
}
```

    - click on the "Stop" button and type the following text into the "Actions" window:

```
on (release)
{
   gotoAndStop("stop");
}
```

- **`Control→Test Movie`**

[4] To add your own labels, simply select a keyframe in the Timeline and type the label text into the Frame Label textbox in the Properties window.

Using Flash's Sound Object to Instantiate & Control Playback

This method for integrating sound into your Flash movies utilizes the capabilities of the **`Sound`** object provided with Flash. In this example, we are going to control sound playback completely through the use of ActionScript, so we need to take a few steps at the outset to inform Flash about our sound file and how we will refer to it. This is accomplished by creating a "Linkage."

- **`File→Open`**, then browse to & select the **"04soundObject_template.fla"** file
- right-click (Windows) or Control-click (Mac) on the name of the sound file ("eurotechno.mp3") in your Library[5] and select "Linkage…" from the pop-up menu that appears
- in the Linkage Properties dialogue box, …
    - type "eurotech" into the Identifier textbox
        - this is the name we will use in ActionScript to refer to this specific sound file; though, in this instance, the Linkage identifier is very similar to the MP3 filename, this is not required
    - place a check in the "Export for ActionScript" box
    - place a check in the "Export in first frame" box

We have now taken all of the preliminary steps necessary to make the sound file accessible to our ActionScript code, so we can begin the process of instantiating a Sound object and attaching a sound file to it, after we accomplish some basic set-up tasks.

- setting up our movie
    - notice that the movie is 10 frames long and consists of three layers (labels, actions, & buttons); note also that there are frame labels on the "labels" layer at frames 1 and 10
        - frame 1: "initialize"
            - the purpose of this frame is simply to carry out some basic initialization steps that only need to be carried out one time at the very beginning of the movie, then cause the playback head to navigate to a later frame where user interaction will occur
        - frame 10: "playback"
- add the initialization code
    - click on the keyframe in frame 1 of the "actions" layer
    - type the following text into the "Actions" window:

```
var currLocation = 0;
var mySound = new Sound();
mySound.attachSound("eurotech");
gotoAndStop("playback");
```

    - allow me to explain the purpose of this code
        - first, I create a variable called **`currLocation`** that is used to track the current position (in milliseconds) within the sound file[6]

[5] Remember, if you do not see the Library window, you can always make it visible by selecting **`Window→Library`** (or **`Control+L`**)

[6] Using this value allows me to cause the buttons to behave exactly as the user expects from her/his experience with CD and DVD players. Notice, as you study the ActionScript for each of these buttons that

  - second, I create a Sound object called **`mySound`**
  - I then use the **`attachSound()`** method to attach the sound file in the movie's Library – remember that we set the Identifier field to "eurotech" when we created the "Linkage" above? – so that any command sent to the **`mySound`** object acts on the "eurotech" sound file
  - finally, I use the **`gotoAndStop()`** command to send the movie's playback head to the "playback" frame label, where all control will be turned over to the user via ActionScript attached to the buttons on the Stage

- add ActionScript to the buttons on the stage to control sound file playback
  - select the Play button and type the following code into the "Actions" window:[7]

```
on (release) {
   mySound.start(currLocation / 1000);
}
```

  - select the Pause button and type the following code into the "Actions" window:

```
on (release) {
   currLocation = mySound.position;
   mySound.stop();
}
```

  - select the Stop button and type the following code into the "Actions" window:

```
on (release) {
   mySound.stop();
   currLocation = 0;
}
```

- **`Control→Test Movie`**

Using Streaming Sound

When you want to ensure that audio-visual synchronization is maintained within a Flash movie, your best bet is to use the "Stream" Sync mode, as described below. To see how this works in a very basic way, follow the steps below. After the general concept is understood, you can take a look at – and get more meaning from – the set of musical form templates available on the author's web site.

- **`File→Open`**, then browse to & select the **"05_I-vi-IV-V_template.fla"** file

---

the "Start" button calls the Sound object's **`start()`** method, using the value of **`currLocation`** (divided by 1000; see next footnote for an explanation) as the starting location. When a user presses the "Pause" button, the current position within the sound file is loaded into this variable before playback is halted, so that when the "Start" button is pressed, playback is initiated from the point where the sound file left off. If, however, the "Stop" button is pressed, the value of **`currLocation`** is set to "0", so that when the "Play" button is pressed, playback starts from the very beginning of the sound file.

[7] There is an inconsistency in the manner in which time is referenced from within the Sound object. All of the objects methods use milliseconds as the unit of time *except* the **`start()`** method which – for some strange reason – uses *seconds* instead!! Beware of this discrepancy when you are creating your own movies that incorporate the built-in Sound object. This is the reason that the **`currLocation`** value must be divided by 1000.

- using the steps followed previously, import a digital sound file of a composition that incorporates the I-vi-IV-V progression into your movie's Library (examples: "Mr. Postman," "Stay," and "Runaround Sue")
- select frame 1 on the "music" layer, then drag the sound file from your movie's Library onto the Stage
- click on frame 1 of the "music" layer a second time, then select "Stream" from the Sync drop-down box in the Properties window
    - notice that the representation of the digital audio file continues from frame 1 until the end of the sound file (which can be *thousands* of frames, depending on the frame rate of your movie and the duration of your sound file)
    - also notice that when you click and drag the playback head (the red rectangle about the Timeline), you can hear the content of the sound file as it passes by
- by creating other layers in your movie to contain objects, animations, interactive elements, you can synchronize, as precisely as you like, the appearance and/or movement of objects on the stage with your audio sound file.

There is a great deal of potential inherent in this method for use in the music classroom, as demonstrated by the form templates available from my web site. Simply point your web browser to: http://www.lecafeamericain.net/faculty.htm, then click on the "stuff" button to navigate to the appropriate page containing a list of available downloads. At the time of NSMIT 2005, Flash templates were available for the 12-bar blues, AABA, and I-vi-IV-V progression.[8] On that same page, you will also find BubbleMachine™, a tool – free to all educators – for the creation of bubble charts to allow point-and-click navigation and interactive exploration of any musical composition that has been saved in MP3 format. The form templates require that the user has *Flash MX 2004* (or later), but the BubbleMachine™ program requires only the free Flash Player. Documentation and basic tutorials are also available from the web site. The author hopes that you will find these templates and tools useful in creating an interactive environment for your students to explore the world of musical sound.

References

Lipscomb, S.D. & Walls, K.C. (in press). Multimedia authoring. In S. Watson (Ed.), *TI:ME Technology Guide*. Hal Leonard Publications.

Webster, P.R. (2002). Computer-based technology and music teaching and learning. In R. Colwell & C. Richardson (Eds), *The new handbook of research on music teaching and learning*, pp. 416-439. NY: Oxford University Press.

[8] There is also a template for the Sonata Form that was created in *Director* available for download from the same web page. More templates will be added, as time allows.