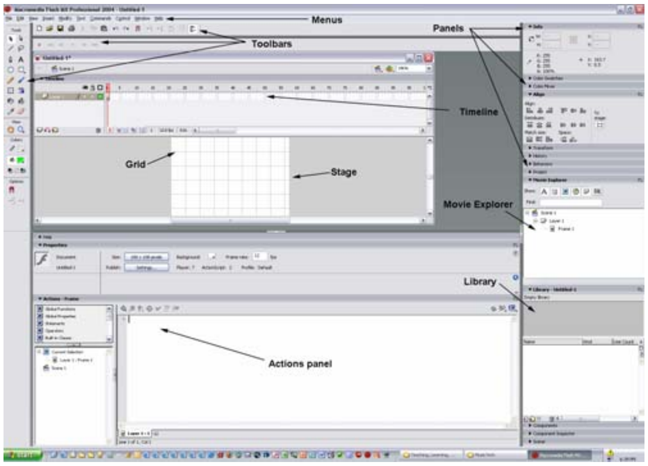
Abridged Proceedings Submission

Using sound in Flash to enhance the music learning experience

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

Presentation made June 17th, 2005
Hartwick College, Oneonta, NY

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 capabilities – for creating various types of quiz forms: true-false, multiple choice, fill in the

---

[1] The opening paragraphs 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*, 2nd edition.

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:

- 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

---

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

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

Animation Basics

Creating Keyframes

- **`Insert→Timeline→Blank Keyframe`** – create brand new content in 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[3]

Importing Sound & Using It in Your Movie

(use: 01soundCrash_template.fla)

- **`File→New`**, then select "Flash Document" from the "General" tab contained in the New Document dialog window that appears
- **`File→Import→Import to Library`**, then browse on your hard drive to select a sound file

---

[3] All templates used for this presentation are available from the author's faculty 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.

- `Window→Library` (or `CTL+L`)
    - notice the sound file has been added to your movie as a Symbol
    - 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
- **Recommended**: add a separate layer labeled "sound"
- IMPORTANT: click on the placeholder frame where you want your sound file to play, then select `Insert→Timeline→Keyframe`
- 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 appropriate layer of the timeline, *drag sound file from Library 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
    - learn about the different "Sync" types (Event, Start, Stop, & Stream) to ensure that you get the most out of using sound in your movies
- `Control→Test Movie`

Adding Sound to a Button
(use: 02soundButtons_template.fla)

- `File→New`, then select "Flash Document" from the "General" tab contained in the New Document dialog window that appears
- 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
- Double-click on the button to enter "Symbol Edit" mode; alternatively, you can select `Edit→Edit Symbols`
    - Just below the labels on the left 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
- Decide which states (up, down, and/or over) to which you wish to attach sounds
    - IMPORTANT: If the frame where you intend to insert the sound does not contain a keyframe, you must add one now (`Insert→Timeline→Keyframe`)
- `File→Import`

  - import an appropriate sound for each of these states
- Add a 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 frame representing the state to which you wish to add a sound, then drag the appropriate sound from the Library
  - repeat this step for every state for which you wish to add a sound
- **`Control→Test Movie`**

- To create a button that will allow the user to stop sound playback …
  - select an appropriate button from the Common Library (or create your own)
  - drag an instance of the button from your movie's Library to the stage
  - click on the Stop button instance
  - add the following code to the "Actions" window:

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

  - **`Control→Test Movie`**

Control a Sound Using Frame Navigation

(use: 03playSound_frameNavigation_template.fla)

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→New`**, then select "Flash Document" from the "General" tab contained in the New Document dialog window that appears
- import the sound file you wish to play into your movie's Library (**`File→Import→Import to Library…`**)
- select a "Play" and "Stop" button from the Common Library or create your own
- create a "button" layer, select the keyframe in frame 1 of that layer, then place an instance of each button on the Stage
- create an "actions" layer, select the keyframe on frame 1, then type the following into the "Actions" window to avoid automatic playback of your movie, turning complete control over to user interactions:

```
stop();
```

- create a "labels" layer and insert two frame labels that will be used to navigate within your movie
  - click on the keyframe in frame 1 of the "labels" layer and type "stop" into the Frame Label textbox in the Properties window
  - insert a keyframe (**`Insert→Timeline→Keyframe`**) at frame 15 of this same layer and type "play" into the Frame Label textbox in the Properties window
  - before proceeding, insert a frame (**`Insert→Timeline→Frame`**) at frame 15 on both the actions & labels layers

- add a “sound” layer to your movie and add a *keyframe* (`Insert→Timeline→Keyframe`) at frame 15 of this layer
- 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 the frame containing this keyframe 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 the frame containing this keyframe 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:
    - 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`

Using Flash’s Sound Object to Instantiate & Control Playback
(use: 04soundObject_template.fla)

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→New`, then select “Flash Document” from the “General” tab contained in the New Document dialog window that appears
- import the sound you want to play into the movie’s Library (`File→Import→Import to Library…`)
- right-click (Windows) or Control-click (Mac) on the name of the file in your Library and select “Linkage…” from the pop-up men that appears
    - remember, if you do not see the Library window, you can always make it visible by selecting `Window→Library` (or `Control+L`)
- 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
    - 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 now 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
    - create three layers and name them: labels, actions, & buttons
    - insert a frame (**`Insert→Timeline→Frame`**) at frame 10 on the actions & buttons layers
    - insert a *keyframe* (**`Insert→Timeline→Keyframe`**) at frame 10 on the "labels" layer
    - using the same technique described previously, add the following frame labels to:
        - 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[4]
            - 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

---

[4] 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 below that the Start button calls the **`start()`** method using the value of **`currLocation`** (divided by 1000; see below) as the starting location. When a users 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.

        “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
  - add buttons to the stage for controlling sound file playback
    - select a Play, Pause, and Stop button from the Common Library (or create your own
    - select the keyframe in frame 1 of the “buttons” layer and drag an instance of each button to the desired location on the Stage
    - select the Play button and type the following code into the “Actions” window:[5]

```
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
(use: 05_I-iv-IV-V_template.fla)

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 my web site.

- `File→New`, then select “Flash Document” from the “General” tab contained in the New Document dialog window that appears
- import the sound you want to stream into the movie’s Library (`File→Import→Import to Library…`)
- in the movie’s Timeline, create a new layer and name it “sound”
- select frame 1 on the “sound” layer, then drag the sound file from your movie’s Library onto the stage

---

[5] 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.

- click on frame 1 of the “sound” layer a second time, then select “Stream” from the Sync drop-down box in the Properties window
- now insert a frame (`Insert→Timeline→Frame`) at frame 60 of the “sound” layer
    - 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 “stuff” 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.[6] 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.

---

[6] There is also a template for the Sonata Form that was created in *Director*, if you are interested.