Intermediate Flash 8

NPPA 2006 Photojournalism Summit / Saturday, June 24, 2006 / Tampa, Fla.
Mindy McAdams / University of Florida / http://flashjournalism.com/
Flash Journalism: How to Create Multimedia Journalism Packages (Focal Press, 2005)

This workshop will cover making multi-part presentations that work well and load quickly. Also: Loading sound files (MP3s) into Flash. If you skipped the "Beginning Flash 8" workshop, be advised that making and scripting control buttons will not be covered in detail in this workshop.

Part 1: Beyond the Simple Slideshow

Examples: http://www.robertowestbrook.com/vukovar/
http://www.heraldsun.com/heart/

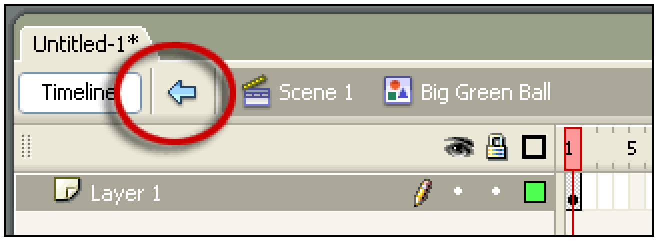
1. Dividing content:
    a. Photo stories
    b. Graphics, maps
    c. Credits / About this
    d. Resource links
2. The opening: The user's first impression
    a. What's this about?
    b. Why should I care?
    c. What are these options or pieces? (What can I do here?)

Part 2: Use of "scenes" in Flash

3. Makes a long Timeline easier to edit and work with, by breaking it up into segments
    a. Makes it easy to change the *order* of segments
    b. Requires fewer layers in the Timeline
4. Creating scenes
    a. Use the Scene panel (Window menu > Other Panels > Scene)
    b. Use icons at bottom of Scene panel to create a new scene, or *duplicate* any scene
    c. In the Scene panel, drag scenes to change the order
    d. Click to rename any scene
5. Viewing and editing scenes
    a. Use the Scene panel, or …
    b. For quick access, use the Edit Scene menu (icon at top right of Timeline)
6. Use scenes in combination with buttons and frame labels
    a. Allows "random access"
    b. Can provide user control—at end of an animated sequence, user decides when to go forward (or back); can allow time to read text or captions

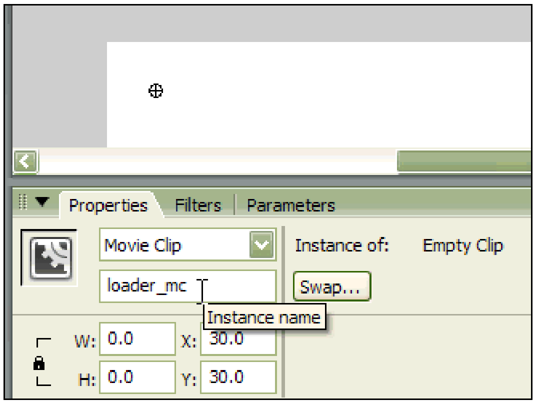
Part 3: Loading external SWFs

7. Empty movie clip(s): The container for external content in Flash
    a. Insert menu > New Symbol
    b. In the dialog box, give it a name (e.g. "Empty Clip") and select Type: Movie Clip. Then click OK.
    c. You are now *inside* the symbol. *Do not stay there!* To return to normal editing, press Ctrl-E (Mac: Cmd-E)—E for "Edit."
    d. In the Library panel, you should now see the new symbol.



IMPORTANT! If you see a *blue arrow,* as shown above, you have double-clicked and gone *inside* a symbol. (In this case, you are inside a symbol named "Big Green Ball.") *Watch out for this*—it is a dangerous mistake! To return to normal editing mode, when you see the blue arrow, press Ctrl-E (Mac: Cmd-E)—E for "Edit." The arrow will be *gray* when you are in normal editing mode.

8. Place an instance of your empty movie clip on the Stage and name it.
    a. Drag it from the Library. Drop it on the Stage.
    b. To use ActionScript to control something, the thing must have an "instance name." When the movie clip is selected, you can name it in the Properties panel (see below). Instance names must not include spaces or punctuation.



Movie Clip
Instance of: Empty Clip
loader_mc
W: 0.0 X: 30.0
H: 0.0 Y: 30.0

    c. Now you can use ActionScript to load external files into that named movie clip instance.
9. Button scripts to load and unload SWFs
    a. If you build each section of a package as a separate SWF, users can load each one as needed, without waiting for them all to download.

   b. Using scripted buttons (see below), one empty movie clip, and external SWF files, you can build a large package in a modular manner.

      ```
      on (release) {
          loader_mc.loadMovie("intro.swf");
      }
      ```

      Displays a SWF named *intro.swf*

      ```
      on (release) {
          loader_mc.loadMovie("maps.swf");
      }
      ```

      Replaces *intro.swf* with a SWF named *maps.swf*

10. Scripts to disable buttons (for an overlay, e.g. credits)
    a. If you want to show some content *on top of* the content that's already visible, don't allow the underlying buttons to remain functional.
    b. *Disable* the buttons with a script.
    c. First, you'll need to give *instance names* to each button (see 8b above). For example: *intro_btn*

       ```
       on (release) {
           intro_btn.enabled = false;
       }
       ```

       Put this script on the button that *opens* the overlay—makes the Intro button not functional.

       ```
       on (release) {
           intro_btn.enabled = true;
       }
       ```

       Put this script on the button that *closes* the overlay—restores the functionality of the Intro button.

Part 4: Use of external MP3 files

11. One MP3, not sync'd
    a. Let's say you have an MP3 file named *music.mp3* … in the same folder with the FLA and SWF files that will use it.
    b. You can load it with two lines of script (below). It will begin playing automatically with this script.
    c. To let the user stop the MP3, and start it at will: Get two buttons from the Flash Common Libraries (Window menu > Common Libraries > Buttons); drag the two buttons to the Stage.
    d. Write script on each button to play and stop, as shown below.

Where is the script?	What is the script?	What does the script do?

On Frame 1	`var x = new Sound();`
	`x.loadSound("music.mp3", true);`
		Associates the MP3 file with the variable “x” and loads it

On a Play button	`on (release) {`
	`x.stop();`
	`x.start(0, 0);`
	`}`
		Makes the MP3 start to play (first, make sure it is stopped)

On a Stop button	`on (release) {`
	`x.stop();`
	`}`
		Makes the MP3 stop playing

12. Multiple MP3 files, synched
    a. This is a bit cumbersome, but easier to script than the next example (below), so it’s better for beginners.
    b. Make a *separate* MP3 file for each photo.
    c. You will load each MP3 as needed, with a script like this:

```
x.onSoundComplete = function {
   x.loadSound("secondfile.mp3", true);
   gotoAndPlay("secondphoto"); //here "secondphoto" is a frame label
}
```

13. One MP3, sync’d (using an array for time markers)
    a. This is what Joe Weiss’ *Soundslides* does automatically for you.
    b. To make this work in a Flash file that *you* create, you will need to get fairly comfortable with ActionScript (sorry, but it’s true).
    c. The basic idea is to check (in your audio editing software) and see *how many seconds have elapsed* in the audio file when you want the first photo to change … then the second photo, the third, etc. For example, 10 photos = 10 timing points.
    d. When you have recorded the numbers for all of your timing points, you will put them in an *array* in ActionScript. An array looks like this:

```
var a = new Array(.01,.35,.59,.95,1.37,1.98);
```

    e. You can download a FLA file that demonstrates this from: *http://www.flashjournalism.com/book/Lesson08/index.htm*
    The *link* is at the bottom of the page, labeled: “loadSound / timing with audio (14) BONUS.” (Make sure to download the MP3 file too!)

    This use of an array is advanced stuff, but if you decide to learn ActionScript, you can master it!