Using Flash with Shockwave

Phillip Kerman

Shockwave can now incorporate Flash 5 content, and that means new authoring opportunities that combine the best of both worlds. Find out the best ways to marry the capabilities of Flash 5 and Shockwave. Where to use Flash in your Shockwave projects • Sending messages between Shockwave and Flash • Optimizing performance

Annotated presentation: **www.phillipkerman.com/mmww/**

__Definitions:

Shockwave is simply a Director movie (.dcr) played through the Shockwave player (or "plugin"). Flash creates .swf files which can are viewed through the Flash Player. Marketing wizards have re-acronymized "swf" to stand for "small web format" (instead of the old "**s**hock**w**ave **f**lash" which confused the matter).

Director and Flash can create both standalone projectors (.exe) or (with the help of a plugin) movies that are viewed in a browser (.dcr or .swf respectively). When viewing a movie through a browser certain potentially harmful or non-applicable "standalone only" features are disabled. For example, Director can overwrite any file. Director has vastly more features that interact with the user's system so the list of off-limit features for web delivery is longer than Flash's.

__Why use Shockwave / why use Flash:

Since the Flash player is practically ubiquitous, and because the previous version of Shockwave (8.0) has penetrated less than 50% of the market (see **www.macromedia.com/software/player_census/**), you'll reach the largest audience by using only Flash. However, Director offers significant features which—if you need them—means it's probably worth requiring web users to install the Shockwave player (or just distribute your application as a standalone Director projector).

Director's unique features include:
—real-time 3D rendering with hardware acceleration;
—support for practically any media type including QuickTime, RealMedia, and externally linked audio or image files;
—fast performance displaying raster graphics;
—memory management (to further improve performance);
—real-time pixel level image manipulation;
—ability to write scripts to aid in production (where Flash's scripting can only execute during runtime);
—more advanced scriptable audio controls;
—limitless extensibility through third-party (or homemade) Xtras;
—and (for projectors) direct access to the user's system and any outside application.

Flash is pretty cool too! Scalable graphics, keyframe animation, and a modern programming language make Flash very attractive. Actually, Flash 5's ActionScript language is arguably better than Director's Lingo. The good news is that with Director 8.5 you can easily include Flash media. Generally, Flash's small file sizes and small plugin come at the expense that most computing resources are offloaded to the user's machine. This makes sense when you consider that computer processors have improved at a much faster rate than connection speeds—even when broadband is considered. Generally, Shockwave movies perform much better than Flash. The bad news is that when Flash movies play inside Director performance degrades to a level below either tool—it's as though the performance hit is cumulative.

To summarize the choice between Flash and Shockwave: use Flash when you can. Director is appropriate when you require special features. Director containing Flash movies is really the ultimate solution—just consider the performance enhancement recommendations below.

__Basic "how-to":

1. Create a Flash movie;
2. Export a .swf (no, you can't play .fla files inside Director);
3. Import it into Director (as either linked or internal);
4. In Director, use the Flash member like any other media by placing it on stage and setting properties as needed.

A note about properties:
Both the Flash cast member as well as each sprite have properties. For example, the Flash member has a "looped" property (set to 0 or 1 depending on whether you want the movie to loop). An individual Flash sprite on stage has an "actionsEnabled" property to control whether actions will be ignored or not, for example. Just like any member, changes to properties affect every instance in the movie. Sprite properties, however, affect only the individual sprite instance.

By the way, you can quickly find all the available member and sprite properties by typing the following into the message window:

```
member("flashMember").showProps()
```
--provided a Flash member named "flashMember" is in your cast

```
sprite(1).showProps()
```
--provided a Flash member is in sprite channel 1.

Don't ask too much:
While Director supports Flash amazingly well, there are some general guidelines worth following regarding how you structure the overall architecture. That is, there are a million features you *can* use—you just don't want to use them all.

Generally, it's best to let Director do all the heavy lifting. Instead of having a complex Flash movie playing inside a complex Director movie, it's best to leave the Flash movie as plain as possible. Offload the programming into Director instead of using Flash. There are exceptions, such as Flash's ability to manipulate and create XML data (something that Director doesn't do). However, it's best to think of Flash as a dumb media type and not as another level of complex programming.

Along the same lines, it's best not to use audio in your Flash movie. When you see how Flash can trigger events in Director (below) you'll also learn how Flash can tell Director to play a specific sound. Frankly Director's sound support is better anyway, so this shouldn't be a big deal. The problem is that inconsistent results will crop up if you have a lot of audio in a nested Flash movie.

You'll see other performance enhancing tricks below, but many come down to reducing the Flash movie to its most simple elements and letting Director control things.

__Interaction between the Director and Flash:

Just because Director will be doing most of the "work" it doesn't mean messages can't be sent between Flash and Director. There are three general categories of messaging and control to consider.

1 Flash to itself: Perhaps pressing a button inside Flash jumps to another frame or changes properties of contained clips. Nothing special here, it's just a matter of learning how to make Flash perform using simple or complex ActionScripting. Indeed it appears that Flash 5 ActionScript is fully supported in .swfs embedded in Director 8.5.

2 Flash sending a message to Director: Perhaps when a Flash animation begins you want to tell Director to start playing a sound. There are two ways Flash can talk to Director. One way ("lingo:") is for Flash to invoke a handler or global event (captured in Director's movie script). The second way ("event:") is for Flash to send a message to the sprite containing the Flash member (which is captured in a behavior attached to that sprite). The difference is subtle.

"lingo:"

Inside Flash, to trigger a Director movie script use:

```
getURL("lingo:someHandler");
```

Where "someHandler" has been defined in the Director movie script.

For example, if in the first frame of a movie clip you have the script:

```
getURL("lingo:playMusic");
```

And in the movie script of Director you have:

```
on playMusic
     puppetSound "musicMember"
end
```

Then Director will execute the puppetSound line which plays an audio member named "musicMember".

In addition to invoking homemade handlers, this method also allows you to trigger any built-in Lingo command. For example, this ActionScript will invoke the Lingo command **beep** inside the host Director movie:

```
getURL("lingo:beep");
```

"event:"

Flash can send a message directly to the sprite where it's residing inside Director. This way Flash can talk directly to a behavior script attached to the same sprite (like Director's **sendSprite()** messaging technique).

For example, if you placed this code in the last frame of a Flash movie clip:

getURL("event:moveIt");

Then if the sprite containing the Flash movie had a behavior attached that read:

```
on moveIt me
     num = me.spriteNum
     sprite(num).locV = sprite(num).locV+10
end
```

The result would be that the entire Flash movie's sprite would move down 10 pixels every time the movie clip reached its last frame.

The "event:" technique simply gives you access to behaviors (instead of movie scripts) which can be generic and self-referencing.

Finally, you can also trigger built-in events in the sprite using this technique. For example, if the Director sprite had this behavior attached:

```
on mouseEnter me
  sprite(me.spriteNum).blend=50
end
on mouseLeave me
  sprite(me.spriteNum).blend=100
end
```

Then, in the Flash movie you placed this script on a button instance:

```
on (rollOut) {
    getURL ("event:mouseLeave");
}
```

The result is that even though the mouse would normally need to move off the entire Flash movie to trigger the **mouseLeave** event (changing the blend back to 100), simply rolling off the Flash button will trigger the same event.

3 Director controlling Flash: Director can access and change properties of the Flash movie. For example, Director could resize the Flash rectangle giving the user a closer look. Or, you might want Director to resize just one clip contained inside the Flash movie.

Consider that Director can control some aspects of the Flash movie that only apply when playing inside Director such as the "directToStage" property which specifies whether other sprites can layer on top of the Flash movie or not. There are also "Flash-centric" controls, like jumping to a particular frame (Flash's **gotoAndStop()**). In any case, Director gets access to these properties and methods using one of the following forms:

```
sprite(n).propertyOrMethod
member("name").propertyOrMethod
```

Where, **n** is an integer containing the sprite channel; **"name"** is the Flash member's name; and, **propertyOrMethod** is the particular property or command to which you want access. If you want to change a property you'd follow the above forms with **=newValue**. Also realize some methods require that you supply additional parameters—like in the case of **setVariable()** you'll need to supply the variable name and its new value.

While I'd love to document each and every feature available using this technique, I'll instead save some examples for the presentation. Remember that you can see all the available properties using the **showProps()** command mentioned above. The only funky thing is remembering which properties are available to members and which are sprite properties. (Use the Behavior Inspector—in list mode—to display all available properties and to see which ones are set-able)

Finally, methods are listed under "Flash" in the "Categorized Lingo" feature in any script window. Of course the help files document them all, but this is a quick and easy way to find them.

General explanation of properties and methods:
Properties include "hair color", "height", and "tooth count". Director sprites have a "locH" property where Flash sprites have the "_x" property. You can access properties and change many of them. "Methods" are more involved. Think of "brushing your teeth" or "combing you hair" as methods. Flash has "play()" and "gotoAndStop()" methods. It's interesting to understand the difference because methods often require parameters included in parentheses that follow the method name.

Note about sending messages to Flash. There's no way to directly trigger Flash events from Director. For example, you can't invoke a Flash function or trigger Flash's **mouseDown** event (without actually clicking the mouse that is). However, you can combine some of the Flash asset methods (mentioned above) with some ingenuity to create the same effect. For example, Director can set Flash variables (through the sprite method **setVariable()**). Therefore, you could have a Flash script (in an **enterFrame** event) that waits for that variable to change and then have Flash trigger its own event. Things are a little bit better with the three newly added commands **call()**, **tellTarget()**, and **print()**. Just realize that there are a limited number of ways Director can affect the Flash movie.

Passing values: Values passed between Flash and Director are always in string form. You can simply place values between quotation marks or, in the case of variables, change them to strings using the String() function. If you want to include quote marks in the values passed to Director be sure to use **\"** which inserts a literal quote.
For example:
getURL("lingo:alert(\"hi mom\")");
will effectively execute the Lingo: **alert ("hi mom")**.

Director's literal quote mark is simply **QUOTE**. For example, the Director code:
sprite(1).setVariable("myVar","you're "&QUOTE&"cool"&QUOTE)
will set a variable (called **myVar**) in the Flash movie to read:
you're "cool"

__Performance enhancement recommendations:

The list of tricks you can try to improve the performance of Flash movies ranges from the theoretical to the superstitious. The following are pretty undisputable tips—but in the live presentation I'll have a more complete list.

1. Start with the best performing Flash movie. The age old Flash performance tips will help you here. Such things as avoiding alpha tweens and unnecessary lines; use Graphic symbols when scripting and other benefits of Movie Clips are not needed; avoid "over animating" with tweens when just a few carefully created keyframes will do better; optimize curves to reduce an image's complexity; etc.

2. Set the Flash sprite to "directToStage". The drawback to this option is that you can't place other sprites (from your Director movie) on top of the Flash movie. You also won't be able to use "background transparent" ink either (which, naturally, is another performance hit). The significance of this tip is great but so is the drawback in how you must design your entire project.

3. If the Flash movie has no animation, set the member's "static" property to 1 (use **`member("name").static=1`**). It won't animate but it performs slightly better. Even if you plan to make the Flash movie animate you can leave the static set to 1 provided that you control the movie's playback from Director (like **`sprite(1).frame=sprite(1).frame+1`**). Also, you can switch this property back and forth as needed.

4. Use quality "auto-high" and "auto-low" during periods of animation when the user won't likely see the rough edges it produces.

5. Avoid having more than one Flash sprite on stage at a time. Again, multiple Flash movies running simultaneously creates a big performance hit.

__Misconceptions worth dispelling:

Director movies are larger. Not true.
Flash has limited programming capabilities. Not true.

__Resources, References:

Most of this content is covered in chapter 14 of my book *"ActionScripting in Flash"* (**`www.phillipkerman.com/actionscripting`**)

Additionally, two chapters are dedicated to Director and Flash in Darrel Plant's book *"Special Edition Using Flash 5"* (published by Que).

Finally, the ultimate online resource for Director can be found at: **`www.director-online.com`**