

# Using Flash MX to Make More Interactive Multimedia Sites

## Webbuilder Las Vegas 9 September 2002

### Phillip Kerman

#### Introduction:

Intro animations are just a fraction of Flash's potential. Attend this session for a look at some simple ways to make Flash sites more interactive. You'll learn the process to turn a complex idea into instructions that Flash understands. In addition to technical details, you'll also dissect design considerations for several particularly effective interactive models in order to reach one simple goal: more engaging Web sites.

#### Timeline techniques:

Design animations using traditional frame-by-frame animation techniques instead of tweening. For example, a bouncing ball will look better if you just design several keyframes instead of letting Flash tween between a few frames. Instead of looking like a computer made your animation it can have personality using techniques such as anticipation and overkill.

Use scripted animations instead using up the timeline. You can actually build your animation offline then extract the details of you animation for use in a script. Storing properties such as locations in an array makes it easy to use the enterFrame event to playback the animation using only one frame. This means your timeline is conserved and independent animations won't conflict. Also, you can effectively change the framerate of individual animations.

#### Scripting techniques:

Learning how program doesn't have to be difficult. Scripting is nothing more than writing instructions. When you sort out the exact instructions you want Flash to follow (using your own words—called "pseudo code"), you can then translate to ActionScript one line at a time. To write a script, think about both what instructions you want followed and *when* you want them to be followed. That is, you must determine what event will trigger the scripts you write. There are many events to which your scripts can respond: button events like "press", "release", and "rollOver", movie clip events such as "load" (when the clip first appears) and "enterFrame" (which automatically triggers as many times a second as your frame rate—say, 12 times a second for a framerate of 12fps). Scripts in keyframes don't need to be "wrapped" within an event as they automatically trigger as soon as the playback head reaches the frame.

#### Effective interactive models:

Many of the effective ways to communicate ideas are also easy to program. First we'll look some ways to design your content and then look at how they're programmed.

*Explore model.* This deceptively simple model is particularly effective. The user sees several buttons on which they can click to reveal more detail. The value is that users can access just the information they want instead of following some kind of order. It's important to clearly indicate which area the user is exploring at any given time.

*Reference model* (or "cross reference"). Like the explore model the user can access just the information they want, but here the user needs to specify at least two preferences before they see all the content. That is, they may want to look at Olympic gold medal winners for Swimming in 2000... or Running in 2000... or Swimming in 1996. In this example, the user specifies the year and the event. This is a great model when you have a lot of content that can be categorized.

*Timed models*. This isn't so much one model, but a general suggestion for situations that use timed events. For example, if you use a long preloader be sure that when it's finished you don't automatically play the content but rather give the user some way to indicate they're ready to proceed (such as a button to click on). In this case, they might have gone to get a cup of coffee during the download and they'll miss the content if it automatically plays. Realize that for any timed event, the user may turn away.

*Tradition*. Standard user interface widgets (such as radio buttons and check boxes) are useful because they always perform the same. Generally, you should learn the standard rules. Not so that everyone's work looks that same, but rather when you do decide to break a tradition, at least you understand the reason for the standard and—hopefully—you can justify your divergence.

#### Getting your hands dirty programming:

Again, it doesn't have to be hard. You don't even need to fit the profile of a programmer (drinking soda in a light-deprived dirty office). People often try to do things the hard way—but with programming, the way to make it easy is to first sort out all your ideas into clear instructions, write pseudo-code, then figure out how to translate into ActionScript one line at a time. Having said how easy it is, you'll ultimately have to do some work. There are foundational skills to learn, such as: addressing clips; setting properties; using homemade variables, built in functions, and homemade functions; and then, getting into ActionScript objects. Even though you can't learn it all at once, as you develop you should constantly criticize your own work looking for ways to improve your code. I suppose the primary goal is first to get something working. But you should also strive to develop efficient production techniques including writing code that's easy to maintain. Often after a project is complete you can walk through how you built it and find ways to improve. No matter what your level you always have room to improve!

#### My books:

I've included everything I think is important to the general Flash developer in the book: "Sams Teach Yourself Macromedia Flash MX in 24 Hours" (ISBN: 0672323710).

My scripting book takes a competent Flash user and turns them into a Flash programmer. This book is also suitable as your first programming book though it concentrates on Flash. "ActionScripting in Flash MX" (ISBN: 0735712956).