

ActionScripting in Flash

by Phillip Kerman

Annotated presentation, downloads, and sample chapters from my two books (*Sams Teach Yourself Flash 5 in 24 Hours* and *ActionScripting in Flash*) available at: www.phillipkerman.com/mdaze/

What can Flash scripting do?

Demonstration of two very interactive sites:

www.m-three.com

I programmed the 1999 version of this site (and part of this year's site) for the agency called "Paris France, Inc." in Portland, Oregon, USA. They designed all the layout and most of the structure. Limited "animation" and transitions were also provided. However, most of the site is purely interactive in that nothing happens unless the user first decides it's time. For example, there are 32 separate "video" clips which are accessible from Flash-made drop down lists. Extensive user control is exhibited in the "scrub" feature of the video clips and the zoomed-in feature of the snowboard details section. For me, it was a great chance to see if Flash can do all the kinds of interactive functionality the way something like Director can—and it can!

<http://holiday.415.com/damn/card/>

I had nothing to do with the production of this piece. However, when I first saw it I knew exactly how much work was involved. I've done projects almost identical in Director. Their attention to detail makes this project stand out! Subtle touches make it very usable. For example, everything is always editable... it's not like you place an object and you're done—no, you can pick it back up to modify or delete. If you click the trash can, the currently selected item is deleted. The fact that most people don't notice anything special (or frustrating) about this interface is proof that "415 Productions" spent a lot of time making it work exactly as they intended.

Another (unrelated) site worth checking out: www.flashcan.com

Where and when does Flash scripting take place?

Three places:

The three places where scripts can be placed (keyframes, button instances, and movie clip instances) correspond to the exact moment that you want the script to execute. A keyframe script will be followed right before its frame appears on screen. In the case of buttons and clips, your script is always contained within an event (either mouse event for buttons, or clip events for clips). The point is, for buttons and clips you must specify the event for which you want your script to occur. For example, you may want a button's script to execute when the user *presses* the button--or perhaps, when they *release* the mouse--or even when they simply *rollOver* the button.

Note: scripts are attached to instances of buttons (not the button symbol).

The Actions panel:

The Actions panel lets you attach scripts to keyframes, button instances, or clip instances. Every component to the ActionScript language is listed and categorized on the left. Some scripts--such as the `play()` and `stop()` command are very simple. Others--such as `gotoAndPlay()` require that you provide additional parameters. Namely, `gotoAndPlay()` requires the destination frame to which you want to *go*.

Exactly how do you write scripts in Flash?

Translating:

Programming is nothing more than translating the instructions you want followed into the ActionScript language. This may sound easy enough, but it's not always so simple. However, the first step is to organize your thoughts.

Pseudo-code:

A great way to write scripts is to first start with "pseudo-code". That is, write the instructions using your own words--try to be clear and definite but don't worry about the syntax. Then, continue to refine your pseudo-code by replacing portions that you know how to express using ActionScript. Eventually, your verbose pseudo-code turns into valid code.

(Exactly how do you write scripts in Flash? continued)

Expressions:

There are many language elements to learn to become a good programmer (including such topics as data types, functions, methods, and objects). Perhaps the most critical concept is understanding expressions and statements. An expression is just a portion of a larger piece of instruction. Thus, statements contain expressions. Flash will evaluate (that is, determine the value of) all expressions. A statement like "make the circle just as wide as the square" includes the expression "as wide as the square" which has a value when evaluated.

Variables:

Variables are another critical concept to programming. They offer a safe (yet temporary) way to store data. You can change what's contained in a variable. Every variable has both a name and a value. This way, you can easily access any variable by name (to change it or just see what value is present).

Properties:

Objects have properties. The most familiar object type in Flash are instances of movie clips on stage. Humans have properties like "hair color", "eye color", "weight", etc. Clip instances also have properties. Some properties are visual (like `_x` position or `_width`) others are not (like `_totalFrames`). Realize that you can make up your own properties and they may or may not be visible. Also, each property is maintained uniquely for each instance (that is, each clip could have a different `_x` position).

Targeting:

Targeting (or addressing) is a concept familiar to HTML authors. Similarly, in Flash you'll need to first address any clip that you intend to affect. The hierarchy of nested clips is used when creating animations and when scripting. Just remember you'll address clips by their instance name (not their source symbol name).

Practical examples of programming techniques applied to Flash.

Example 1 Stop Play

This example simply shows how a plain stop() and play() command (attached to buttons) will cause the main timeline to pause or proceed. In the case of the nested clips (of rotating wheels) these clips need to be addressed if they are to stop too: car.frontWheel.stop() for example.

Example 2 Koko and Max (setting properties)

This example shows how the clip instance containing the picture of my dog (Max) will change visually when the script addresses the clip's _alpha property. For example dog._alpha=50 will cause the clip instance named "dog" to be 50% transparent. The up and down arrows are slightly more complex. They don't just change the _alpha to a specific value, rather they increase or decrease the _alpha from its current level. Either: dog._alpha=dog._alpha+10 or dog._alpha+=10 can be used to increase the _alpha percentage.

Example 3 www.m-three.com (excerpt)

This example show first, how the scale properties (_xscale and _yscale) of a clip will change as the mouse moves up and down. When the mouse moves left or right the speed and direction of the clip changes. This involves both ascertaining the mouse's location (_xmouse and _ymouse) as well as executing a script repeatedly (either the clip event enterFrame or mouseMove).

Example 4 Slide Show (from ActionScripting in Flash)

This example (for which you can download the chapter and the source file from www.phillipkerman.com/actionscripting) shows how variables and the enterFrame clip event can be combined for a great visual effect that doesn't require a ton of complex scripting. For every enterFrame event (executed, say, 12 times a second) the clip containing the pictures has its _alpha changed by a value equal to the variable "alphaChange". Naturally, if alphaChange is zero you won't see any change. When alphaChange is set to -5 (when either button is pressed) you begin to see the clip's _alpha reduce. If-statements are used to both change the value of alphaChange (to +5) when the clip's _alpha goes below zero and to set alphaChange (back to 0) when the clip's _alpha goes above 100. More details will be shown in the presentation and can be found in the book.

Things to know

Variables and Children:

Creating variables is not unlike having children. In both cases you simply need to give them a name and then take care of them. In the case of variables there are a few rules for what name is used (for example, no spaces and you can't begin with a numeral). Besides that, you can create variables as needed. Any time you need to store information while the movie runs go ahead and use a variable. As the author you are responsible to maintain the variables by setting their values. Remember this analogy if you are ever afraid to make up a variable as needed.

Variables are really "homemade properties":

You can actually have several instances of the same named variable and there will be no confusion *if* you realize that variables are linked to the clip (or timeline) where they are used. In this way, you can think of variables as homemade properties. Conceptually, they're the same thing (each clip can maintain its own set of properties as it can with variables too). Plus, the syntax to access variables is the same as properties.

Monitoring techniques:

Since you're responsible for maintaining the variables' values, you'll want to learn how to monitor them. The `trace()` command will reveal the value of any expression in the output window when testing. Also, Dynamic Text fields will continually show the value of any variable listed in the Text Options panel.

Additionally you can see the current value for any variable or property by using the Debugger. (No, it doesn't actually "de-bug" your movie for you...it's more of an inspector that lets you view details of your movie as you test it.)

Everything cool is done to a Movie Clip:

Clip instances have many benefits over other symbol types. For example, only clips can be given instance names--which, by itself, is not terribly exciting. However, only clip instances can have their properties accessed (and changed) through scripting. Plus, only clips give you access to clip events (most notably `mouseMove` and `enterFrame`).

The only time to use buttons is when you want to use a button feature (such as the built-in Over and Down states). Only use Graphic symbols when you *need* to preview an animation as you "scrub" the timeline or when you want to use one of the loop settings available only from the Graphic's instance panel.

(*Things to Know* continued)

Functions are your friend:

Copying and pasting *anything* is a cry for help. Perhaps the best way to modularize and make your code more maintainable is to learn how to write your own functions. See chapter 8 on functions from my book, viewable at: www.phillipkerman.com/actionscripting/

But wait, there's more:

I don't know if this is the good news or bad news, but there's so much more you can learn with ActionScript. It's a good language to use because it's based on a great standard (the same as JavaScript). You'll pick up skills that are transferable. It truly is nearly limitless.

Summary

Scripts are nothing more than instructions written in a language Flash understands (ActionScript).

For every script you must decide *when* you want it execute.

Variables and properties are maintained individually per clip instance.

Accessing variables and properties always involves first addressing the clip in which you are interested.

Be relieved that your efforts to learn ActionScript will give you great power (with Flash) and will also transfer to any programming endeavor.