Scripting in Flash
by Phillip Kerman

What can Flash scripting do?

Demonstration of two very interactive sites:

www.m-three.com

I programmed this site for an agency called "Paris France, Inc." in Portland, Oregon. They designed all the layout and most of the structure. The 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 or 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.

Where does Flash scripting take place?

Flash scripts are called Actions. Actions include things like "Stop" "Play" and "GetURL" (which navigates the user to another web page). Think of Actions as functions that *do* something.

Actions go in one of two places: keyframes & button instances. In the case of keyframes, the Action is executed (that is, it takes place) when the playback head reaches that keyframe. Since buttons are symbols stored in the Library, you can have as many instances on screen as you want. "Under" each instance of a button you can place Actions which will execute when the user clicks the button.

Flash has an Expression Editor which provides a reasonable interface to help you write your scripts. For example, all the available functions are listed and easily incorporated without the need to remember all the terms or look them up in a book.

Phillip Kerman +1 503 236-7721 phillip@teleport.com www.teleport.com/~phillip

Exactly how do you write scripts in Flash?

Standard programming terms and concepts can be applied to Flash.

Expressions:

Expressions are the "sentences of code" that you write in any programming language. Expressions can be simple statements that do things (like `username="phillip"` which, translated, means set the variable `username` to now equal the string "phillip"). Expressions can also be comparisons (like `if score>90 then...` which, translated, means if the variable `score` contains a number greater than 90 then do whatever comes next). Expressions can get much more involved combining statements, comparisons, and routines that perform calculations repeatedly. The cool part is, as long as you follow the language's syntax you can write anything you want! (Making it do what you want is another matter.)

Variables:

Variables provide a safe storage location for numbers or words. Think of variables like separate little white-boards. You can write anything you want on a white-board... erase it and write something else in its place. Or, you can write on a white-board and leave it to be viewed later when you need to recall what was written. Variables are not unlike white-boards because you put any thing you want in a variable (like writing on a white board, you can put the string "phillip" into a variable). Each variable has a unique name for two reasons: you need to know which variable you're setting (on which white-board you're writing, if you will); also, variables need a name so you can check what's currently in a particular variable (like checking to see what's written on a particular white-board, you need to know *which* white-board).

Properties:

The easy way to think of properties are as characteristics—like personally, my hair color property is brown, my name property is Phillip, etc.. However properties need not be visual or even based in reality—like a timer could have the "timeOut" property containing the time at which the timer expires. The point is the current value for a property can always be ascertained—as in "what is your age property currently?" Additionally, many properties can not only be ascertained, they can be changed—like "set the x location of that graphic to 50". In Flash practically all properties are visual, all are settable, and are related to Movie Clips only.

Targets:

Since it's likely you'll want your Action Scripts to *do* something (like move a shape to an new location on screen or change an image's size) you must specify to whom you're talking when you say "do it". In Flash this concept is called "Targeting". That is, you target the Movie Clip you want to affect. Luckily, if you understand HTML relative references you can apply that knowledge to Flash. (I bet you never thought that information would come in handy, eh?)

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

Practical examples of programming techniques applied to Flash.

Exercise 1 involves a simple linear animation in which we have a button to stop and one to play. However, this only stops or plays the main timeline. Movie Clip symbols have a timeline of their own—so our "stop" doesn't stop the Movie Clip we've included from playing. In the demonstration, we add another set of buttons that use Tell Target to send an Action to a specific instance of the Movie Clip. After the quick demonstration we view a finished piece using Tell Target extensively.

Exercise 2 includes a button which causes a Movie Clip to change its alpha value. Initially, the button causes the Movie Clip alpha to jump to 50. (In pseudo code "Set the alpha property of the Movie Clip to 50"). Then we make the button increment the alpha to lower and lower values. ("Set the alpha property of the Movie Clip to the current alpha property of the Movie Clip minus 10.) Finally, we add a simple "If" statement to prevent our button from causing the alpha to go lower than 0. ("If the alpha is greater than 0 then go ahead and do the increment trick").

Things to know

Variables:

You have great latitude how you use variables in Flash. In one way, creating variables is like having children... you must name them, then take care of them.

Scope:

Variables "live" in the timeline where they are used—that is their scope is limited to this timeline (they're "local" to that timeline). It's a double-edged sword. On the one hand it's nice because you can have several different variables with the same name and they won't conflict. In this way, a variable can be used *like* a homemade property. "MySpeed", for example could be used inside several different instances of a Movie Clip—and each can contain a different value. The other side of the sword is it can be confusing—as in the case of George Forman's multiple kids named "George Jr.".

So normally, when referring to a variable by name Flash will consider the variable (of that name) in the current timeline only. However, you can precede the variable's name with the "path" to where it resides such as "someMovieClip/someClipInside/". In Flash 4 you just need to separate this path from the actual variable name with a colon (":"). For example, "someClip/:aVariable" refers to "aVariable" which is in the Movie Clip called "someClip". Relative references work the same way as relative links in HTML (e.g. "../" for "one folder up").

(*Things to know* continued)

Monitoring techniques:

Often you'll create variables which are very important to your movie even though they remain "behind the scenes"—effectively invisible. That is, you could be tracking a user's "score" in a variable but that doesn't mean they need to see it at all times. However, as you test your movie it is often nice to "monitor" the variables to assure they're functioning as you expect. The easiest way to monitor your variables is to create a text field (not plain text) and just make sure the Text Field Properties of the field has the variable you're monitoring in the "Variable:" field. One thing to note (and this comes in handy later) the variable will not only be displayed in this field... but if you change the contents of this field you're actually changing the value of the variable. It's effectively "locked" to the field. One little tip: if you're just monitoring variables during testing, a great technique is to keep them in their own layer and just remember to change the layer properties to "Guide" before you deliver—this way the fields won't be displayed to your user and you can quickly restore them for further testing by resetting the layer properties back to Normal.

Literal vs. Expression

Think of Literals as strings that always include quotation marks (the actual characters, if you will) ... think of expressions as formulas which are interpreted as code. Although this is fairly straightforward, Flash seems counter intuitive in one place particularly. In the set variable function you must specify your variable name as a Literal. In all other cases it's pretty clear. Just ask yourself: is it the value of what I'm typing that I want evaluated (Expression) or is it the literal text that I'm typing which I want used (Literal). *Exercise 3* involves the difference between setting the variable counter to equal counter+1 literally or as an expression. If counter+1 is set to literal we see the string "counter+1" put into the variable. However, if we used expression the code counter+1 is evaluated and then our variable increments by 1 each time the Action is executed.

Everything cool is done to a Movie Clip

Practically all properties which are accessible (through Get Property) and almost every settable property (through Set Property) are properties of Movie Clips. Of course there are often other reasons to choose Movie Clips—namely, you want something to animate on its own timeline. Regardless, you want to make sure to use Movie Clips any time you'll be needing to affect a graphic at runtime (e.g. stretch, move, change alpha, drag, etc.).

(*Things to know* continued)

"Call" subroutine

A common programming code structure is a subroutine. Flash 4's subroutine is named "Call". You simply put all your code in a keyframe's Action tab... make sure the keyframe has a Label... then from anywhere in the timeline you can Call that frame (by label name). You don't actually "go" to that frame... you just visit long enough to execute the Actions in that frame. Actually, you probably want to put this keyframe (which is being called) off the end of the timeline beyond where the user will actually visit. A great clue that you should consider using "Call" is when you find yourself copying and pasting code. Even if the code you are duplicating is slightly different in each instance, if you are creative you can still use Call to make your code generic.
Exercise 4 shows how Call can recycle code. Also, a creative way was shown to use the same code (via Call) for both an increment and decrement effect.

Gateway to outside world

JavaScript
Flash can tell JavaScript to execute JavaScript functions.
JavaScript can talk to Flash in one of two ways:

By telling Flash to do something (like "go to frame 10").
By asking Flash for information (like "Flash, on what frame are you currently?")

For more on JavaScript follow links at *www.teleport.com/~phillip/ff2k/*

Form GET and POST
To send variables from your movie use: Get URL

To load variables into your movie use: "Load Variables" (found under "Load Movie")

GET and POST require that there is a script (CGI or similar) "listening" on the server. Your Flash movie can simple tell this script something—like the name of a user login name (and that can be the end of it). Or, you could have the script send variables back to Flash.

Advanced techniques

There are a few invaluable techniques for advanced scripting in Flash. Although these are unlikely to be included in the demonstration, samples files can be found at *www.teleport.com/~phillip/ff2k/*. Additionally, the included article from Macromedia User Journal "Flash and Your Inner Programmer" has detailed descriptions.

All exercises demonstrated are available at: *www.teleport.com/~phillip/ff2k/*