

Advanced ActionScripting with Flash M

Robert Berdan 8/23/02

Materials: bring along Lesson 1 from Text book on floppy or the CD to distribute work file among students. In the future ask students to bring their text book and CD to class. Exercise requires Flash MX – if not loaded, download from www.macromedia.com good for 30 days. First evening – provide overview of course, action script basics, planning and then build the electric light bulb in Chapter 1 and explain the components of the script.

Introduction: The course pre-requisite is previous experience with Flash preferably haven take the Flash course at SAIT. This program will not teach you how to create basic animation it is assumed you already know how to do that. While javascript or C programming is not required – if you have had experience it will be very helpful. To better understand programming in Actionscript – which is based on javascript – a course on javascript or C programming would be very helpful. I will attempt to explain the basics so you have an understanding of the concepts behind the scripts for those who have never programmed, for those that are expert programmers, you will gain some insight into the differences between actionscript and javascript.

Overall Objectives: To show you how to take FlashMX to another level and add interactivity to your movies using Actionscript

- 1) To apply action script in FlashMX in order to create interactive movies
- 2) You will learn how to load data into flash and send it to a simple database
- 3) You will create a standalone business card CD
- 4) You will learn how to load music and pictures dynamically into a flash movie

TextBook – required, many of the tutorials will be carried out using this book, all the files you require are on an accompanying CD.

There will also be exercises and tutorials not on the CD created by Science & Art staff.

Evaluation will based on

- 1) attendance 10%
- 2) assignments 90%

Lecture #1

Objectives:

- 1.1 Understand the benefits of ActionScript
- 1.2 Identify script elements – syntax basics
- 1.3 Plan a project
- 1.4 Write your first interactive script and debug
- 1.5 Test your script

Action scripting basics

Open up Flash MX – lets go to the Action script window and look at the various components. If Actions panel is not visible select window>Actions

- 1) Expand the actions panel by dragging it up
- 2) Identify the various components of the window
- 3) Toggle between normal mode and expert mode
- 4) In Expert mode – select view line numbers
- 5) Select the book icon – help reference for action script
- 6) Identify the find, find & replace buttons, target buttons, +, - buttons to add or remove lines of code
- 7) Circle with cross – insert target path
- 8) View built in actions and index
- 9) Check mark select to check syntax
- 10) Autoformat button (not will not work if you errors in your code)
- 11) View preferences for typing code
- 12) Adding prebuilt in actions double click on action and it appears in window in normal mode, switch to expert mode you can still add lines of code and receive hints
- 13) Note drop down (URL like window) – it can be used to select from several scripts you might have in a movie – just select the script from the drop down menu to edit.
- 14) Up and down arrows move your cursor up and down one line at a time

PIN script – Used to prevent Flash from displaying a different script as you click different items in your movie. Pinning forces Flash to continue displaying the script as you move around your movie or select a different frame. To pin a script the pin button must be changed from its horizontal position to one point down and to the left.

Customizing the Actions Panel

Edit>Choose preferences>Action script editor tab - you may want to change the color of keywords, comments, or modify the font size.

NOTE Flash highlights syntax errors in red and deprecated actions in green. If you set you publish settings to create Flash 5 movies any text that would break in an earlier version will colored yellow.

ACTIONS –TWO TYPES

- 1) Actions added to keyframes
- 2) Actions attached to objects – buttons or movie clips

It is possible to create scripts that target text fields in order to modify or retrieve information from them, but text fields on their own cannot have attached scripts.

It is strongly recommended that you create a separate layer to insert all your actions. This will keep them more organized and make it less likely that you will

add two actions to the same place in a movie. You can lock the layer – this way you will not be able to add graphics, however you will still be able to add actions.

To Add Actions in the script pane you can do it a variety of ways.

1. double click on the action
2. drag the action into the pane
3. right click the action, choose add to Script (right click allows you to select reference to view parameters and other info for any action)
4. You can use shortcuts – all start with Esc key (to view short cuts go to the options menu in the top right corner of the script pane and turn on view shortcuts and they will appear beside the action commands)

The Actions are organized into Categories – if you can find one, you can go directly to the index where all actions are listed alphabetically.

Deprecated Actions

As any computer language evolves and develops, new words are added and old ones are phased out. In the actions toolbox, deprecated actions are gathered together in the Deprecated category – open it and take a look.

E.g. Movie clips used to be controlled using tellTarget – this has not been changed to with

Flash 5

```
on (release) {  
    tellTarget ("cat")  
    {  
        stop();  
    }  
}
```

In Flash MX

```
on (release) {  
    cat.stop();  
}
```

Flash MX player continues to support deprecated actions but it is recommended that the newer commands be learned and used in place of the older ones.

(Good place to take a break and send students to go and pick up their text books if the book store is open).

Script Elements

Elements of action script include words, punctuation, and structure – based on javascript which in turn is based on the C programming language. The program is executed from the top down.

Actions statements with instructions – they are case sensitive

```
MyCost= 5:00;  
CashRegister.gotoandPlay(50) ;
```

Semicolons are used at the end of most statements (but not after loops or conditional structures), they are optional in actionscript and javascript but in some languages they are mandatory (e.g. PHP)

Dot syntax - is a way of attaching objects to properties and methods and describes a hierarchical relationship among the objects.

Cat - object

Cat.black - black is a property of the cat

Cat.run() - run is a method (something the Cat can do)

Cat.black.small - string of properties

Dots act similar to / in file path to show the relationship or target path of a specific timeline.

Eg. _root.movie puts the movie in the root or main movie timeline.

E.g wheel._rotation

Parentheses ()

```
CashRegister.gotoAndPlay(50)
```

Script tells the movie to go to and play frame 50

Value inside the parentheses is a specific value or parameter that can be modified

Quotation Marks

"" or " double or single

used to denote textual data

e.g.

name = "Fred" ; Fred is text

name= Fred;
name=10; name is a number

Indentation and Spacing

Indentation and spacing is ignored for the most part, however indentation makes your script easier to read and is recommended

Curly Brackets or Chicken Lips {}

```
on (release) {  
    MugCost= 5.00;  
}
```

line up the brackets for easier viewing, must always close any bracket you open

Comments

// comments goes here

multiline comments (need to test if OK)

```
/* put comments in here  
another line here */
```

Operators

- 1) arithmetic +, -, /, ^, =, *
- 2) string +
- 3) conditional operators <, >, <=, >=, ==, !=
- 4) logical operators &&, ||, !

&& and, || or, ! not e.g. ! if true false (reverses the logic statement)

Events

Things that occur during playback or that the user does to modify the movie timeline

e.g.

on (release) - on release of the button do something e.g. gotoAndPlay(30)

Variables – storage containers for data, hold data in the computers memory so they can be manipulated, called upon or used in Expressions

setProperty command

var in javascript is a keyword that seems to work just fine

3 Datatypes strings (text), numbers e.g. 10, Boolean values e.g. true or false

Keywords

Break, case, this, typeof, if, else, new, return, while, var, with etc

These are words that in actionscript have special meaning and must not be used as a variable name or your script will have errors (bugs)

Planning a project

When planning a project ask yourself some questions?

What do you want to occur?

What pieces of data do you need to input or track?

What needs to happen in the movie prior to a script being triggered.

What events will trigger the program or script?

Are there decisions to be made when the script is made –i.e. if this occurs do this.

What will your scene look like?

Problem: You want to create a payment system for an electric company – the user can type in the amount paid, the program then compares the amount paid to the amount owed and sends back a message – you paid in full, you underpaid or you overpaid (we will also make a light bulb change from off, to on, to hot depending on the amount paid).

The value owed – will be imported from a database (simple text file – since this value will likely change from month to month).

You may want to start with sketch of the layout or use box diagrams to illustrate (see page 16) for the project we are going to do.

You will need Flash **electricbill1 fla** file from your Flash text book, place into a

folder and open up notepad. You can follow the instructions in the book or the instructor.

1) open notepad and type in `electricBill=60` (text uses `$electricBill=60` significance of `&` not known but is not necessary). You are basically creating a variable called `electricBill` and setting it to a value the user owes, the value can be changed at any time. Save this file into the folder with your movie.fl`a` file and call it **electricBill.txt**. This value will be loaded dynamically into Flash.

2) Open Flash MX and load the movie **electricbill1.fl`a`**

3) Note the various layers in the movie, actions layer, text fields (input text field and dynamic text field), switch (button) and light (movie clip).

4) Select the dynamic text field – amount you owe, and in the properties box give it an instance name **owed**.

Text boxes can be assigned variable names or instance names. Usually you will want to give the text box an instance name. You must give it an instance name if you want to control the text field itself, reposition the box, resize it etc. When you give a box a variable name the box displays the contents of the variable. Using ActionScript to change the text property of the text instance allows you to manipulate and change its contents without using variables.

5) Select the box underneath and give it the instance name **paid**

6) Select the “invisible text area box” at the bottom of the movie clip and give it the instance name **message** – this is where we provide the user with feedback regarding the amount they paid versus how they owed.

7) Select the light bulb movie clip and give it the instance name **light** (if you right click on the light bulb and select edit, you will see the movie clip is made of 3 key frames having the labels Off, On, and Hot - which represent the 3 states of the movie

Now that all the scene elements are named we are ready to start scripting. Open the actions panel set it to Expert Mode, select Frame 1 of the main movie and enter the following script:

```
LoadVariablesNum("Electric_Bill.txt",0)
```

The command tells you to load the text file into the movie clip at level 0 or the main movie. To view the parameters of the LoadVariable command select the it from the index and then select the book icon to see the text below. If you are doing it in Normal mode – inside the Browser/Network folder (Might be good to this step both ways Normal and Expert to see the variable available and levels).

Explanation of Code from Code Index

loadVariables ("url" ,level/"target"[, variables])

Parameters

url An absolute or relative URL where the variables are located. If you access the movie using a Web browser, the host for the URL must be in the same subdomain as the movie itself.

level An integer specifying the level in the Flash Player to receive the variables. When you load variables into a level, the action in the Actions panel in normal mode becomes loadVariablesNum; in expert mode, you must specify loadVariablesNum or choose it from the Actions toolbox.

target The target path to a movie clip that receives the loaded variables. You must specify either a target movie clip or a level (level) in the Flash Player; you can't specify both.

variables An optional parameter specifying an HTTP method for sending variables. The parameter must be the string GET or POST. If there are no variables to be sent, omit this parameter. The GET method appends the variables to the end of the URL and is used for small numbers of variables. The POST method sends the variables in a separate HTTP header and is used for long strings of variables.

Description

Action; reads data from an external file, such as a text file or text generated by a CGI script, Active Server Pages (ASP), or PHP, or Perl script, and sets the values for variables in a Flash Player level or a target movie clip. This action can also be used to update variables in the active movie with new values.

The text at the specified URL must be in the standard MIME format application/x-www-form-urlencoded (a standard format used by CGI scripts). The movie and the variables to be loaded must reside at the same subdomain. Any number of variables can be specified. For example, the phrase below defines several variables:

```
company=Macromedia&address=600+Townsend&city=San+Francisco&zip=94103
```

The first movie to open in an instance of the Flash Player loads into the bottom level (identified in code as `_level0`). When you use the loadMovie or loadMovieNum action to load subsequent movies into the Flash Player, you must assign a level number in the Flash Player or a target movie clip into which each movie will load. When you use the loadVariables action, you must specify either a Flash Player level or a movie clip target into which the variables will load.

Example

This example loads information from a text file into text fields into the varTarget movie clip on the main Timeline. The variable names of the text fields must match the variable names in the data.txt file.

```
        on(release) {
            loadVariables("data.txt",
                "_root.varTarget");
        }
```

8) Now select the 10 frame in the actions layer insert keyframe and add the following code

```
owed.text = electricBill;
stop();
```

The reason we add this to keyframe 10 is to provide some time for the text file to load (frame cushion) – if you don't do this and someone clicks on a button before the file finishes loading your script may respond incorrectly.

The command basically outputs the value owed into the text field we gave the instance name **owed**. If you change the value in the text file, the new value will load into this box electricBill is a variable inside the electric_Bill.txt file we created and set the value=60. (see page 21 in text)

Text fields are considered objects in Flash MX and one of its properties is **text**

9) Now select the light Switch button and the following code

```
on (press) {
    amountPaid = Number(paid.text);
    amountOwed = Number(owed.text)
}
```

Note the Number() function has limitations it can not change Number("dog") into a number it will return a value isNAN (is not a number) because a numeric conversion here is not possible.

When the button is pressed, the text values displayed in the paid and owed text fields are converted to numbers and they are assigned variables named amountPaid and amountOwed.

10) With the actions panel still open, add the following lines of script – Conditional statements that determine what the script should do when different values paid are typed in (page 23). Add the code within the curly braces.

```
on (press) {
    amountPaid = Number(paid.text);
    amountOwed = Number(owed.text)
}
```

```

if (amountPaid < amountOwed)
{
    difference = amountOwed - amountPaid;
    light.gotoAndStop("Off"); // Off is the frame label
    message.text = "you underpaid your bill by "+
difference +"dollars";
}

else if (amountPaid > amountOwed)
{
    difference = amountOwed-amountPaid;
    light.gotoAndStop("Hot"); // hot is the frame label
    message.text = Your overpaid your bill by " +
difference + " dollars";
}

else
{
    light.gotoAndStop("On")
    message.text = "You have paid your bill in full"
}

}

```

Save the file and test your movie by selecting Control>Test movie (fla file must be stored on HD not the CD for this to work since flash creates a temp file). Try typing in different values – the try typing in a text message – what happens?

You will need to another line of code to check for whether or not the use has entered text instead of a number. The command **isNaN()** is a built in function which checks if the value entered is a number.

11) At the top of the if statements add the following code.

```

if (isNaN(amountPaid))
{
    message.text = "Please enter a proper dollar amount";
}

```

This script needs to be at the top so it is used first and you must add else if before the other if statement.

Summary

- 1) You have familiarized yourself with the Action script window
- 2) Planned and developed a simple action script project
- 3) Written and tested your script
- 4) Fixed bugs to complete the project