Outline for

UCLAE Flash 1: Session #6

|  |  |
| --- | --- |
| TITLE | Basic Interactivity |
| DATE | Thursday 7pm – 10pm, July 31th, 2008 |
| INSPRIRATION | Right Brain (Drawing & Animating) vs. Left Brain (ActionScript) - <http://www.funderstanding.com/right_left_brain.cfm>  Simple Math Test: What the difference between 60,000 and 40,000? |

1. INTRODUCTION

Administrative issues

* The answer to the ‘Inspiration’ question above – $20,000. Right, well $60,000 is the average starting of a Flash position for ActionScript fluent candidates, $40,000 is the average starting salary for those who know no ActionScript. So learn ActionScript, really really well.(Salaries are gross approximations of Los Angeles, CA, USA, 2008).
* Take the quiz for session #6
* Show “Vector vs. Raster” example on blackboard from Session #5

1. OUTLINE

Chapter 12. Building Buttons for Interactivity

* Key Concept: A FLA 9 FLA can be published for ActionScript 1.0, 2.0, or 3.0. This is best set when you first create a new Flash document. We will learn only ActionScript 3 since it is the latest and greatest. Avoid confusion by avoiding learning 1 or 2 unless your job requires it. Many teams are still using 2 as they are still learning 3.
* Key Concept: ActionScript can be placed many places; on symbol instances, in symbol instances, on frames, in an external document class, or in an external class. For now we will place ALL ActionScript on frame 1 of the main timeline. Best practice is to place ALL ActionScript in external classes.
* Key Concept: If we want our Flash movie to behave the exact same way every time, we need no ActionScript. Otherwise (applications, games, websites, media players, interactive animations, etc…) we will need ActionScript. Even if using ActionScript not part of your primary role, knowing enough ActionScript to collaborate with those who do, will make you far more valuable.
* Creating a Basic Button Symbol
  + Insert -> New Symbol (Choose Type: Button)
  + Create 4 states by drawing on each of the first four frames
    - Up – Shown while mouse cursor is not over the button
    - Over – Shown while mouse cursor is over the button and the mouse button is pressed down
    - Down – Shown while mouse cursor is over the button and the mouse button is pressed down
    - Hit – Non-visible, this defines where the mouse cursor must be to ‘hit’ the button.
  + Return to main timeline
  + Drag Button instance from library onto frame 1 of a new layer
  + Publish your movie and interact with the Button instance via the mouse
* Teachback: Earn (1) point to repeat ‘Creating a Basic Button Symbol’
* Creating Shape-Changing Button Symbols & Creating Fully Animated Button Symbols
  + Place a MovieClip instance on any of the visual states; Up, Over, or Down state. Each will play when shown.
* Components & Using Button Components & Modifying Button Components
  + Components are premade buttons and other interactive controls. They can save you time so you don’t have to ‘recreate the wheel’ for common things such as buttons, scrollbars, textInputs, and more…
  + Window -> Components -> User Interface
  + Select a component instance on the stage, Window -> Component Inspector
  + Components are advanced in nature and unless you choose to use them, they are unneeded for this class.
* Key Concept: Creating Button instances is fast and easy to create 3 state buttons. If we need more than 3 states, we’ll need to create Button MovieClips. First let’s create a 3-state button but using a Button MovieClip.
* Creating Movie-Clip Buttons
  + Insert -> New Symbol (Choose Type: MovieClip)
  + Create a new layer called “Labels” and select it
  + Create a keyframe on each of the first four frames
  + Select each of the 4 frames one at a time, use the Property panel’s ‘Frame’ label input box to enter these names.
    - \_up
    - \_over
    - \_down
    - \_hit
  + Create a new layer called “Art” and select it.
  + Create 4 states by drawing on each of those four frames (just like you did for the button)
  + Return to main timeline
  + Drag MovieClip Button instance from library onto frame 1 of a new layer
  + Publish your movie and watch each frame play
  + In the next chapter we’ll see how to make a MovieClip Button behave like a Button via ActionScript.
* Key Concept: Frame labels must be unique in a given Flash document. Otherwise an error will be shown upon publishing your movie.
* Key Concept: The Buttons and Button MovieClips we created react to our mouse but do little else on their own. We’ll learn how to apply ActionScript next.

Chapter 13. Basic Interactivity

* I will distribute today session notes through the Mac OS. Using Copy-and-paste on ActionScript will be helpful today.
* Key Concept: ActionScript is a series of commands, or statements, that makes Flash perform tasks at runtime – that is when a published Flash (SWF) file runs in the Flash Player for your end users to view.
* Key Concept: Your Flash Movie will automatically play the main timelines frames and the frames of any MovieClips. We’ve seen that Flash will automatically toggle the state of a button under the mouse. If much more interactivity is neede, ActionScript must be added to your movie.
* Key Concepts: Terms
  + Similar Terms: ActionScript = Code = Actions = Statements
  + Variable or Property: An ActionScript structure that holds a value or values
  + Data Type: Defines what range of values is acceptable for a variable.
  + Function or Method: An ActionScript structure that executes one or more lines of ActionScript.
  + Scope: Location where a statement will execute. E.x. (‘on a button’ vs. ‘on frame 1 of the main timeline’
  + Target Path: The ‘address’ of an instance on the Stage. E.x. A named MovieClip ‘clipB’ inside another named MovieClip ‘ClipA’ on the stage would have an absolute target path of ‘clipA.ClipB’ (Note the ‘.’ between each instance name. Target paths are important because that is how you ‘call’ a variable or a function that exists on a specific instance.
  + Case Sensitivity: This is a characteristic of ActionScript 3.0 and many other programming languages which means that using upper or lower letter case when typing code does matter, so be careful when following examples.
  + Comment: Text which is not executed, it is a note for humans to read. Statements that begin with “//” are comments. Statements, on one or more than one line that begin with “/\*” and end with “\*/” are also comments. These are called ‘multiline comments’ since they usually define many lines at once as comments.
  + Syntax: The logical use of numbers, letters, and punctuation to execute ActionScript. Saying an ActionScript statement has good syntax is like saying an English language statement has good grammar and spelling.
* Key Concepts: Punctuation & Syntax (See Textbook Page 371 for definitions) Examples;
  + Dot ‘.’
    - main\_button.visible
  + Semicolon ‘;’
    - main\_button.visible = false;
  + Colon ‘:’
    - var score : Number = 10;
  + Braces ‘{}’

function mouseDownHandler(event:MouseEvent): void {

trace ("Mouse Down on Button");

//trace ("Scope: " + this.parent);

}

* + Brackets ‘[]’
    - var list : Array = [1,2,3];
* Touring the Actions Panel
  + Open the Actions Panel, Window -> Actions
  + Expand the Actions Toolbox and the Script Navigator by clicking the small arrow on the left edge of the Script Pane
  + Explore the Actions Panel with the mouse, watching for rollovers which explain each button
* Output Window (Window ->Output) and Compiler Errors (Window -> Compiler Errors)
  + The Output Window shows the results of any Trace statements. Trace statements are never seen by the end user. Trace statement output exists to help the developer (you) to test the code as it is being run.
  + The Compiler Errors Window will show you any MAJOR errors in your code. It is possible for your Flash movie to publish without showing any compiler errors, but still not function properly. If you don’t get the desired results, inspect your code closely.
* Adding Frame Actions, Traces, and Event Handlers to your Flash Movie
  + Create a New Movie
  + Create a new layer and name it shapes
  + Create a keyframe on frame 1,2, and 3, and draw a unique shape in each frame. This will be useful so we can tell which of the three frames the movie is on at any time.
  + Create a new layer and name it “Button”
  + Drag the Button instance from the library of your last movie or create a new button.
  + Select the Button instance. In the Properties panel name the instance ‘main\_button’
  + Create a new layer and name the layer ‘Actions’
  + Select the first frame
  + Open the Actions Panel
  + Type the following ActionScript the Actions Panel window

//FRAME ACTION

stop();

//gotoAndStop(2);

//gotoAndPlay(3);

//TRACING

trace ("Hello From Frame 1");

//trace ("Scope: " + this);

//trace ("Scope: " + main\_button);

//trace ("Scope: " + this.main\_button);

//EVENT HANDLER

main\_button.addEventListener(MouseEvent.MOUSE\_DOWN, mouseDownHandler);

function mouseDownHandler(event:MouseEvent): void {

trace ("Mouse Down on Button");

//trace ("Scope: " + this.parent);

}

* Customizing the Actions Panel
  + Flash -> Preferences (Mac) or Edit -> Preferences (PC)
* Organizing Frame Actions (Best Practices)
  + Keep all the Actions on your Timeline in one layer
  + It IS significant on which frame you place your code. Code on Frame 10, for instance, runs when the play-head reaches frame 10, etc…
* Adding Frame Actions
  + Select a Frame
  + Open the Actions Panel, Window -> Actions
  + Add the desired ActionScript
* Programming Buttons with Frame Scripts & The IF Conditional-Statement
  + Create a new layer called “Labels”. On frame 2 add a keyframe and use the Properties panel to label it ‘clicked\_frame’
  + Replace the big block of Actions above with the following;

//FRAME ACTION

stop();

//EVENT HANDLER

main\_button.addEventListener(MouseEvent.MOUSE\_DOWN,mouseDownHandler);

function mouseDownHandler(event:MouseEvent): void {

var x : Number = 2;

if (x == 2) {

gotoAndPlay(“clicked\_frame”);

} else {

//do nothing

}

}

* Previewing Actions at Work
  + Publish your movie
* Modifying and Extending Button Scripts
* Choosing Events
  + Replace the big block of Actions above with the following;

//FRAME ACTION

stop();

//EVENT HANDLER

main\_button.addEventListener(MouseEvent.MOUSE\_DOWN,mouseDownHandler);

function mouseDownHandler(event:MouseEvent): void {

trace ("mouseDownHandler");

}

main\_button.addEventListener(MouseEvent.MOUSE\_UP,mouseUpHandler);

function mouseUpHandler(event:MouseEvent): void {

trace ("mouseUpHandler");

}

main\_button.addEventListener(MouseEvent.ROLL\_OUT,mouseRollOutHandler);

function mouseRollOutHandler(event:MouseEvent): void {

trace ("mouseRollOutHandler");

}

//THERE ARE MANY MANY MORE EVENTS FOR BUTTONS AND OTHER THINGS TOO...

* Using One Event Handler for Multiple Events
  + Duplicate ‘main\_button’ by copy-and-paste
  + Rename the duplicate as ‘second\_button’
  + Replace the big block of Actions above with the following;

//FRAME ACTION

stop();

//EVENT HANDLER

main\_button.addEventListener(MouseEvent.MOUSE\_DOWN,mouseDownHandler);

second\_button.addEventListener(MouseEvent.MOUSE\_DOWN,mouseDownHandler);

function mouseDownHandler(event:MouseEvent): void {

if (event.target == main\_button) {

trace ("main\_button mouseDownHandler");

} else {

trace ("second\_button mouseDownHandler");

}

}

* Button Components
  + The premade Button component works like the button you’ve already created.
  + Remove the ‘second\_button’ from the Stage
  + Open the Components Panel (Window -> Components -> User Interface)
  + Drag a Button to the stage
  + Name the instance ‘second\_button’
  + Rerun the code. It will work just as before.
* Scripting Movie Clips to Act As Buttons & Linking to Other Web Pages
  + Remove the ‘second\_button’ instance from the Stage
  + Create a Button MovieClip
  + Drag an instance to the Stage
  + Name the Instance ‘second\_button’
  + Replace the big block of Actions above with the following;

//FRAME ACTION

stop();

//BUTTON ACTION

second\_button.buttonMode = true;

//EVENT HANDLER

main\_button.addEventListener(MouseEvent.MOUSE\_DOWN,mouseDownHandler);

second\_button.addEventListener(MouseEvent.MOUSE\_DOWN,mouseDownHandler);

function mouseDownHandler(event:MouseEvent): void {

if (event.target == main\_button) {

trace ("main\_button mouseDownHandler");

} else {

trace ("second\_button mouseDownHandler");

//OPEN A BROWSER WINDOW

var urlRequest : URLRequest = new URLRequest ("http://www.google.com");

navigateToURL (urlRequest, "\_blank");

}

}

* Teachback: Earn (1) point to create a new file and use a Button Component instance to trace an event.
* Using Buttons to Control Timelines
  + Let’s do a demo!
* Using Buttons to Control Graphic-Objects
  + Let’s do a demo!
* Transforming Timeline Animations into Code
  + Create a new Flash Movie
  + Create a Tween of any kind
  + Select all frames of the layer containing the tween
  + Edit -> Timeline -> Copy Motion As ActionScript 3.0 (Choose ‘newInstance’ for the name in the dialog box)
  + Create a new Layer called ‘ActionScript’ and select the first frame
  + Paste within the Actions Panel (Code will be pasted)
  + Create a new Layer
  + On the new layer, create a new Symbol Instance called ‘newInstance’
  + Publish and see ‘newInstance’ animate

NOTES

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_