Controlling Visual Objects with ActionScript

In this unit, you will learn how to control visual objects on the Stage with ActionScript.

2006 Adobe Systems Incorporated. All Rights Reserved.

Objectives

After completing this unit, you should be able to:

- ▶Reference visual objects in ActionScript
- ▶Set properties in ActionScript to control visual objects
- ▶Customize the focus rectangle on a mobile device
- ▶Understand how to reference object paths in ActionScript 2.0
- ▶Dynamically create `MovieClip` instances

2006 Adobe Systems Incorporated. All Rights Reserved.

Adobe

Adding Actions to Timeline Frames

In Flash Lite 2, you can add ActionScript in two places inside the FLA:

- ▶Attached to an instance of a `Button` or `MovieClip`. These actions run when an event related to the instance occurs.
- ▶Attached to a frame in the Timeline. These actions run when the playhead reaches the frame.
- ▶Adding code to one Timeline frame - usually the first frame of the main Timeline - is the preferable practice.
- ▶Most important, grouping all code together facilitates collaboration among multiple developers, since these developers do not have to sort through "noodle code" spread across dozens of locations in a FLA document.

Note: Later units will discuss the use of external class files for code organization

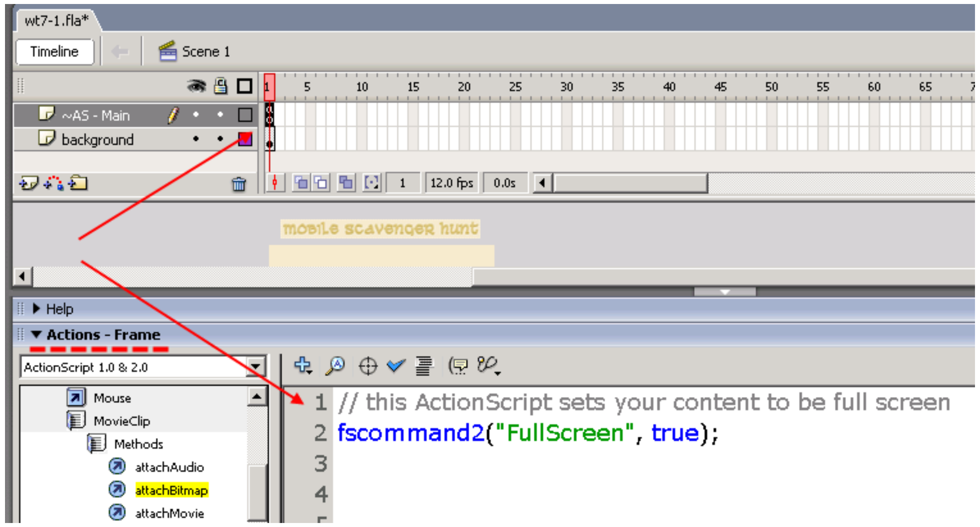
2006 Adobe Systems Incorporated. All Rights Reserved.

Adobe

Adding ActionScript using the Actions panel

To add ActionScript to a document:

1. Create a new layer on the main timeline, above all other layers
2. Select the first frame
3. Open the Actions panel
4. Write ActionScript within the Script Pane



```
// this ActionScript sets your content to be full screen
fscommand2("FullScreen", true);
```

Figure 1: Adding ActionScript to the main timeline

2006 Adobe Systems Incorporated. All Rights Reserved.

Adding ActionScript using the Actions panel

Keyframes with attached ActionScript are marked with a lowercase 'a' symbol. The attached code will execute when the playhead plays that frame.

Note: ActionScript is usually added to the top-most layer, as the visual assets it controls are normally loaded into the Flash player beginning from the bottom-most layer.

2006 Adobe Systems Incorporated. All Rights Reserved.

Adobe

Controlling visual objects

Objects in ActionScript 2.0 can be broadly grouped into visual and non-visual categories.

- ▶Visual classes are classes whose objects appear visibly on the Stage.
- ▶Non-visual classes have no visual element and are generally created and used only through ActionScript.
- ▶For example, `TextField` is a built-in visual class of the Flash Lite Player. When you draw a `TextField` on the `Stage`, you are instructing the Flash Player, through the SWF, to create a `TextField` object based on the `TextField` class, and place it at a particular `x` and `y` position.
- ▶Every `MovieClip`, `Button`, and Input or Dynamic `TextField` property can be assigned or changed at runtime through ActionScript.

2006 Adobe Systems Incorporated. All Rights Reserved.

Controlling MovieClips and TextFields

An instance name is a name for an object on the Stage. Specifically,

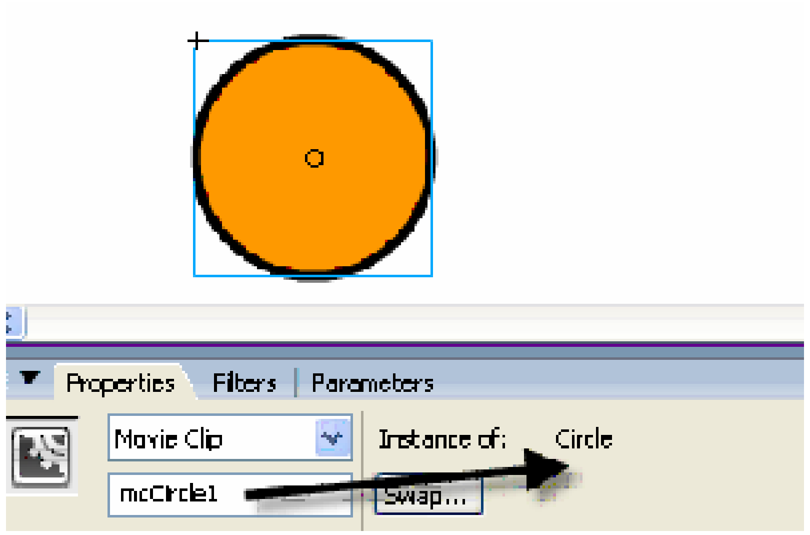
- ►a `MovieClip` symbol instance,
- ►a `Button` symbol instance,
- ►or an Input or Dynamic `TextField.`

You use the instance name assigned in the Properties panel (or in your code) to "talk" to that object in ActionScript.

2006 Adobe Systems Incorporated. All Rights Reserved.

Assigning instance names for visually created objects

In the illustration below, the instance name `mcCircle1` has been assigned to an instance of a `MovieClip` symbol named `Circle`.



Movie Clip
Instance of: Circle
mcCircle1

Assigning an instance name

2006 Adobe Systems Incorporated. All Rights Reserved.

Assigning instances names in ActionScript

- ▸When writing ActionScript, the best practice is to declare your instance names, just as you declare any variable.
- ▸This means you will both assign an instance name in the Properties panel, and declare it in your code. By doing this, you get code hinting in the Actions panel.

```
var mcCircle1:MovieClip;
```

- ▸ActionScript also has objects that have no visual counterpart on the stage. You'll see this kind of object created and assigned an instance name through code:

```
var dtToday:Date = new Date();
```

2006 Adobe Systems Incorporated. All Rights Reserved.

Setting properties using ActionScript

- ▶A property is a variable attached to a visual or non-visual object, which contains information describing some characteristic of that object.
- ▶You've already assigned property values to visual objects using the Properties panel. If you know an object's instance name, you can assign any of its properties with ActionScript.
- ▶To do so, type the instance name, followed by a dot operator (.), then the property name. After that, you assign a value to that property using the assignment operator (=):

```
instanceName.propertyName = value;
```

- ▶If you assigned a data type to your instance name when you declared it, the Actions panel will give you code hinting after you type in the dot operator.

Note: For legacy reasons, some properties begin with an underscore (_) character. Use them when required.

2006 Adobe Systems Incorporated. All Rights Reserved.

Writing text into a TextField

`TextField` objects expose properties to control all aspects of their visual and non-visual behavior. For example, to place text in a `TextField` object use the following code:

```
  txtProdName.text = "Great Game!";
```

2006 Adobe Systems Incorporated. All Rights Reserved.

Setting the position of a MovieClip

- ▸MovieClips object similarly expose a wide range of controllable properties.
- ▸For example, to position a MovieClip object, set its _x and _y properties use the following code:

```
mcBall3._x = 15;
mcBall2._y = 40;
```

2006 Adobe Systems Incorporated. All Rights Reserved.

Walkthrough 1: Controlling Visual Objects

In this walkthrough, you will perform the following tasks:

- ►Add the `navBar` to the application
- ►Set the text property on the `TextField` soft key labels
- ►Create the `join` view

2006 Adobe Systems Incorporated. All Rights Reserved.

Creating Visual objects at runtime

New `MovieClip` symbol instances can be created at runtime using the `attachMovie()` method.

There are several methods for creating visual objects at runtime:

- ▶Create all `MovieClip` objects at design time and place them offstage, then move them onstage as needed at runtime using ActionScript. However, this may unnecessarily consume memory.
- ▶Create visual objects, then hide and reveal them using the `_visible` property of the `MovieClip` class. This should be avoided in Flash Lite development, unless essential, as non-visible objects are still included in the player's scanline rendering calcuations, and may impact performance.
- ▶Attach `MovieClip` objects from the Library as they are needed. This is the best practice approach.

2006 Adobe Systems Incorporated. All Rights Reserved.

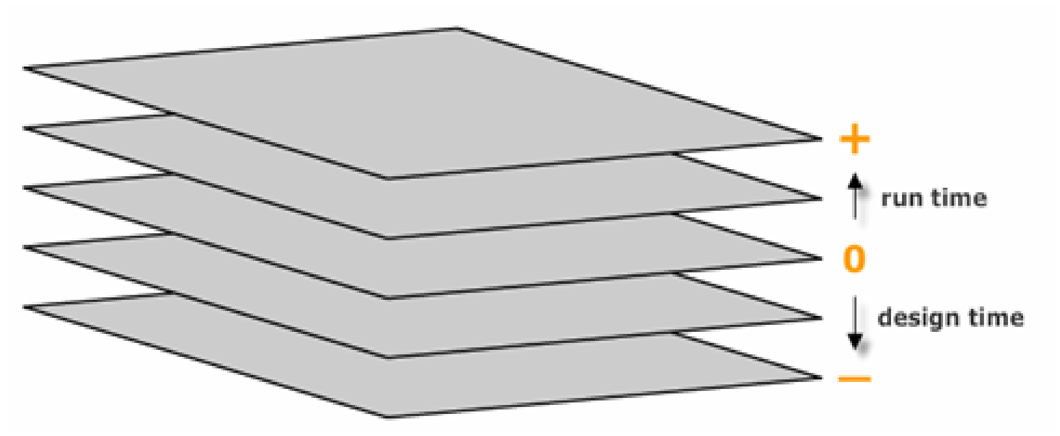
Understanding Movie Clip Depth

- ▸The "depth" of a `MovieClip` object is an integer from -16,383 to 16,382 that represents its stacking order - its position along the z-axis - of its parent `MovieClip`.
- ▸Only one controllable visual object may reside at any given depth.
- ▸Depth values are assigned automatically at design time by the Flash 8 Pro authoring tool. Values are assigned based on the relative order of the layer on which a visual object is placed.
- ▸Individual objects' depth may also be controlled using tools on the Modify > Arrange menu item. Objects with higher values appear "closer" to the eye, and objects with lower values appear behind other objects, if covered.

2006 Adobe Systems Incorporated. All Rights Reserved.

Understanding Movie Clip Depth

Figure 2: MovieClip Depth



2006 Adobe Systems Incorporated. All Rights Reserved.

Controlling object depth using ActionScript

`MovieClip` objects expose three methods for use in assigning and controlling depth values using code.

- ▶`getNextHighestDepth()`: this method returns the next higher unoccupied depth within the current MovieClip on which it's invoked.
- ▶`getDepth()`: returns the depth of the current object on which it's invoked.
- ▶`swapDepths(target|value)`: exchanges the target object's depth with the current object. Alternately, you may specify the depth to which the current object should be set as an integer.

2006 Adobe Systems Incorporated. All Rights Reserved.

Using attachMovie() to create visual objects

To create a `MovieClip` symbol instance at runtime:

1. Create the `MovieClip` symbol
2. Export the symbol for ActionScript
3. Assign a Linkage Identifier
4. In ActionScript, create an instance of this symbol using `attachMovie()`
5. In ActionScript, configure the newly created instance

2006 Adobe Systems Incorporated. All Rights Reserved.

Making a symbol available for runtime creation

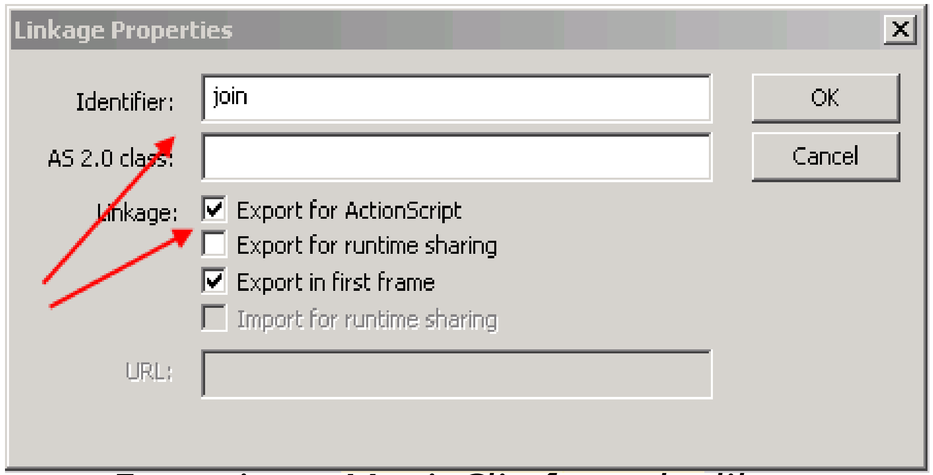
- ▶`MovieClip` symbols must be exported and assigned a Linkage Identifer, before they can be created at runtime.
- ▶The `attachMovie()` method will fail if used with a un-exported symbol.

2006 Adobe Systems Incorporated. All Rights Reserved.

Exporting a symbol for ActionScript

To export a symbol for use by ActionScript:

1. Right-click over the symbol in the Library
2. Select Linkage ..
3. Check Export for ActionScript
4. Notice the Identifier matches the symbol name. Do not change it.



Linkage Properties

Identifier: join

AS 2.0 class:

Linkage: ✓ Export for ActionScript; ☐ Export for runtime sharing; ✓ Export in first frame; ☐ Import for runtime sharing

URL:

Exporting a MovieClip from the library

Note: If you uncheck the Export in First Frame checkbox, you must place an instance of the symbol on the Stage in a Timeline and keyframe that will run before the symbol is used or it will not be included in the compiled SWF file.

2006 Adobe Systems Incorporated. All Rights Reserved.

Creating symbol instances at runtime

- ▶The `attachMovie()` method can be called globally, as a method of this current `MovieClip` timeline in which it's called on a keyframe, or as a method on another `MovieClip`.
- ▶The new instance will be a child inside whatever `MovieClip` on which the method is called.
- ▶The basic syntax for the `attachMovie()` method of the `MovieClip` class is:

```
// create on this current timeline
attachMovie("Linkage Identifier", "new Instance Name", depth)
-or-
// create within another clip in this timeline
parentMovieClip.attachMovie("Identifier", "instanceName",
  depth);
```

2006 Adobe Systems Incorporated. All Rights Reserved.

Creating symbol instances at runtime

The following code would create a new runtime instance of a `MovieClip` symbol exported with the Linkage Identifer `login`:

```
attachMovie("login","login_mc", getNextHighestDepth());
```

Note: `getNextHighestDepth()` *ensures a unique depth, but no other properties are initially set. So the instance will appear as though its* `x` *and* `y` *are both set to zero..*

2006 Adobe Systems Incorporated. All Rights Reserved.

Adobe

Assigning initial properties of a MovieClip created with ActionScript

- An optional fourth parameter may be passed to the `attachMovie()` method.
  - This parameter, called an initialization object, is a generic object that holds values to assign to corresponding properties of the new symbol instance being created.
  - Those may be built-in properties of the `MovieClip`, or application-specific data properties to hold information related to the newly created clip.

2006 Adobe Systems Incorporated. All Rights Reserved.

Creating an initialization object

- ▸ActionScript supports a type of complex variable called a "generic object".
- ▸It is simply an instance of the `Object()` class.
- ▸Any property desired can be attached to an `Object` object, using either dot notation, or object literal notation.
- ▸To directly instantiate an object, use this syntax:

```
var init:Object = new Object();
init._x = 15;
init._y = 45;
init.screenName = "Login";
```

Alternately, you may use object literal syntax, to create an identical object:

```
var init:Object = {_x:15, _y:45, screenName:"Login"};
```

Object literal syntax may be used anywhere an object variable is needed. However, it can create somewhat difficult to read code.

2006 Adobe Systems Incorporated. All Rights Reserved.

Using an explicit initialization object

Any properties assigned to an initialization object will be assigned to a newly created instance when the object is passed as the (optional) fourth argument to the `attachMovie()` method.

For example:

```
var init:Object = new Object();
init._x = 15;
init._y = 45;
init.screenName = "Login";

attachMovie("login", "login_mc", getNextHighestDepth(), init);
```

Note: While `_x` *and* `_y` *are properties of the* `MovieClip` *class,* `screenName` *is an invented property, to hold custom information specific to the example application*

2006 Adobe Systems Incorporated. All Rights Reserved.

Using a literal initialization object

While this approach may quickly lead to hard to read code, object literal syntax can be used inline in the `attachMovie()` method. In this example, the new clip would be positioned at an `_x` of `15` and `_y` of `45`.

```
attachMovie("login", "login_mc", getNextHighestDepth(),
  {_x:15, _y:45, screenName:"Login"});
```

2006 Adobe Systems Incorporated. All Rights Reserved.

Using initialized data

- ▶Initialization object properties set the default values of the new instance, if they are actual `MovieClip` properties.
- ▶Otherwise, they are simply available as a data properties of the new instance, useable by other code. For example:

```
// display the current screen name
currentScreen_txt.text = logic_mc.screenName;
```

2006 Adobe Systems Incorporated. All Rights Reserved.

Controlling attached movie clips

Following initialization, programmatically created movie clip instances can be controlled through either the instance name passed to the `attachMovie()` method, or through the reference returned when calling this method.

2006 Adobe Systems Incorporated. All Rights Reserved.

Assigning the instance name by method

The second argument to the `attachMovie()` method of the `MovieClip` class becomes the instance name of the newly created movie clip.

```
attachMovie("login", "login_mc", getNextHighestDepth(),
  {_x:15, _y:45});
login_mc._x += 10; // move 10 pixels down
```

2006 Adobe Systems Incorporated. All Rights Reserved.

Adobe

Assigning the instance name by returned reference

The `attachMovie()` method returns a reference to the newly created instance, which can be assigned to a previously created variable.

```
var login_mc:MovieClip;
login_mc = attachMovie("login", "", getNextHighestDepth(),
  {_x:15, _y:45});
login_mc._x += 10; // move 10 pixels down
```

Note: When using this approach, you must still pass either the same name as the second argument to `attachMovie()`*, or pass an empty* `String`*. You cannot omit the second argument to this method.*

2006 Adobe Systems Incorporated. All Rights Reserved.

Walkthrough 2: Organizing Visual Content

In this walkthrough, you will perform the following tasks:

- ►Create a user defined function to attach a MovieClip
- ►Add a MovieClip to the view using attachMovie()

2006 Adobe Systems Incorporated. All Rights Reserved.

Dynamically Controlling MovieClips

ActionScript code may be used to generate instance names, positions, and any other property of a programmatically created `MovieClip`.

2006 Adobe Systems Incorporated. All Rights Reserved.

Creating an indefinite number of instances

The `attachMovie()` method may be used when an unknown number of movie clips must be added. In this case, you may use a `for` loop to loop over the number of needed movie clips attaching them while giving them unique instance names on the fly.

2006 Adobe Systems Incorporated. All Rights Reserved.

Creating an indefinite number of instances

The code below loops 7 times to attach seven uniquely named and positioned instances of the `product` movie clip symbol from the Library to a `MovieClip` named `grid_mc`:

```
var numOfProducts:Number = 7;
var instanceName:String;
var xPos:Number = 10;
var yPos:Number = 20;
for (var i:Number = 0; i < numOfProducts; i++)
{
 instanceName = "product" + i + "_mc";
 grid_mc.attachMovie("product", instanceName,
  getNextHighestDepth(), {_x:xPos, _y:yPos});
 trace(instanceName);
 // displays product0_mc, product1_mc, etc.
 xPos += 10; // increment _x by 10 for each
}
```

2006 Adobe Systems Incorporated. All Rights Reserved.

Adobe

Removing movie clips at runtime

The `removeMovieClip()` method, when called on a `MovieClip` with a positive depth value, deletes the instance from the Stage, and removes it from memory. It accepts no parameters and returns no value:

```
currentScreen.removeMovieClip();
```

Note: Instances with a negative depth must be swapped to a positive depth before removal, or the similar `unloadMovie()` *method may be used.*

2006 Adobe Systems Incorporated. All Rights Reserved.

Adobe

Walkthrough 3: Dynamically Controlling Views

In this walkthrough, you will perform the following tasks:

- ▶Create variables to track active `MovieClip`
- ▶Create a method to return a dynamic instance name
- ▶Dynamically attach `MovieClips`

2006 Adobe Systems Incorporated. All Rights Reserved.

Adobe

Summary

- ▸Any `MovieClip`, `Button`, or `TextField` property may be controlled at runtime using ActionScript.
- ▸A property is a variable attached to an object, which describes a characteristic of that object.
- ▸To control an object, it must be assigned an instance name.
- ▸ActionScript may be attached to keyframes, `Button` symbol instances, or `MovieClip` symbol instances.
- ▸The best practice is to attach ActionScript to keyframes.
- ▸ActionScript executes when the playhead plays its keyframe.
- ▸Every `MovieClip` instance has a unique depth value.
- ▸Higher depth objects cover lower depth objects.
- ▸Visually created objects have negative depth values.
- ▸`MovieClip` instances may be programmatically created using the attachMovie() method of the MovieClip class.

2006 Adobe Systems Incorporated. All Rights Reserved.

Summary

- The `getNextHighestDepth()`, `getDepth()`, and `swapDepths()` methods let you determine and control the depth of a MovieClip instance.
- Instances of a `MovieClip` symbol exported for ActionScript with a Linkage Identifier may be programmatically created using `attachMovie()`.
- Programmatically created `MovieClip` instances:
  - must be assigned a unique depth value, generally determined using `getNextHighestDepth()`.
  - can be configured as they are created using an initialization object.
  - can be controlled after creation through their instance name.
  - can have dynamically generated instance names.
- Programmatically created `MovieClip` instances are removed from Stage and memory using the `removeMovieClip()` method.

2006 Adobe Systems Incorporated. All Rights Reserved.