BEST PRACTICE

CONTROL VIDEO PLAYBACK

Find out how to load, stream and control *Flash* video using ActionScript to create a clever zoetrope effect

SOFTWARE: FLASH 8

Over the last few years, *Flash* has been making a slow but steady march towards, among other things, providing a platform-free, quality method of delivering video. Macromedia, and now Adobe, have done this by introducing embedded video in *Flash MX 2004*, and then improving it with *Flash 8*.

Flash Video has become more and more ubiquitous, and now sites such as YouTube.com, panjea.com and video.google.com have played a big part in bringing the format into mainstream acceptance.

Here you will learn how to use ActionScript in *Flash 8* to load, progressively stream and control multiple video clips to create a zoetrope effect – a device designed to simulate moving imagery from a series of still images by spinning a cylinder that is decorated with slots that allow the viewer to see inside as it spins.

Within the cylinder is attached a series of still images, each image part of a moving sequence, much like a flipbook. As the viewer looks through the slots, the pictures appear to blend into a moving image. In some ways, this is similar to the way modern-day movie projectors work.

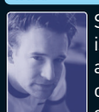
ON THE CD

You'll find all the files you need to complete this tutorial in the folder marked Tutorial\Part on this issue's CD.

TIME NEEDED

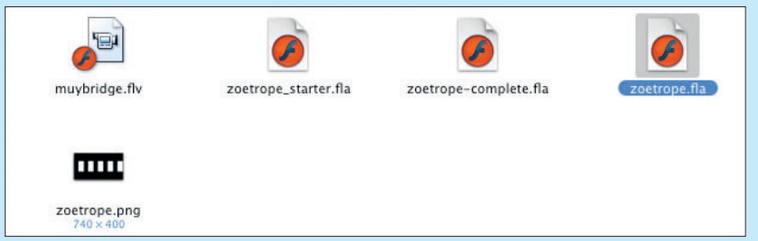
30 minutes

INFO



Shane Rebenschied is a fan of old paper and stains, and can often be found staring at pieces of corroded, rusted metal. He is a professional freelance illustrator, author, *Flash* designer, developer and consultant, and lives in the Arizona desert. Check out his online portfolio at www.blot.com.

USE ACTIONSCRIPT TO LAUNCH VIDEO



1 Duplicate the file zoetrope_starter.fla, provided on the CD, and rename the duplicate zoetrope.fla. In this starter file, the faux zoetrope animation, the video objects (placeholders where the video will be displayed), the video (FLV) that you'll be loading, and an ActionScript layer have already been created. Here you'll write the ActionScript to make it all work.

```
1  // video 1
2
3  var myNetConnection1:NetConnection = new NetConnection();
4  myNetConnection1.connect(null);
5
6  var myNetStream1:NetStream = new NetStream(myNetConnection1);
7  video1.attachVideo(myNetStream1);
```

2 Open the zoetrope.fla file and select the first keyframe in layer A. Open the Actions panel (Window→Actions) and type the following:

```
// video 1
var myNetConnection1:NetConnection = new NetConnection();
myNetConnection1.connect(null);
var myNetStream1:NetStream = new NetStream(myNetConnection1);
video1.attachVideo(myNetStream1);
```

This code will create new NetConnection and NetStream objects, which will handle the loading, playback and controlling of your first video clip. The last line attaches the NetStream object you just created to one of the pre-made video objects on the stage named "video1".

```
1  // video 1
2
3  var myNetConnection1:NetConnection = new NetConnection();
4  myNetConnection1.connect(null);
5
6  var myNetStream1:NetStream = new NetStream(myNetConnection1);
7  video1.attachVideo(myNetStream1);
8
9  myNetStream1.onStatus = function(infoObject:Object) {
10     if (infoObject.code == "NetStream.Play.Stop" && infoObject.level == "status") {
11         myNetStream1.seek(0);
12         myNetStream1.pause(false);
13     }
14 }
```

3 The videos will play and stop, so you'll need code to tell *Flash* to play the video again when it reaches the end. Click after the last line of code, hit Return twice, and then type:

```
myNetStream1.onStatus = function(infoObject:Object) {
if (infoObject.code == "NetStream.Play.Stop" && infoObject.level == "status") {
                    myNetStream1.seek(0);
                    myNetStream1.pause(false);
          }
}
```

```
myNetStream1.onStatus = function(infoObject:Object) {
    if (infoObject.code == "NetStream.Play.Stop" && infoObject.level == "status") {
        myNetStream1.seek(0);
        myNetStream1.pause(false);
    }
}

// ======================================= \\
```

4 To make it easier to differentiate between the blocks of code in your Actions panel, add a comment that serves as a break. Create two line breaks after the last line of code and type:

```
// ======================================= \\
```

The two foward slashes will tell the ActionScript parser that everything on the line after it is a comment and should not be executed when the movie is played.

5 Click at the end of the line you just created and add two line breaks beneath it. Now, select all the ActionScript you've written in the Actions panel so far. Copy that ActionScript, click in the last empty line break you just created at the bottom of the Actions panel, and then Paste the ActionScript to that location.

```
myNetStream1.onStatus = function(infoObject:Object) {
    if (infoObject.code == "NetStream.Play.Stop" && infoObject.level == "status") {
        myNetStream1.seek(0);
        myNetStream1.pause(false);
    }
}

// ================================== \\

// video 1

var myNetConnection1:NetConnection = new NetConnection();
myNetConnection1.connect(null);

var myNetStream1:NetStream = new NetStream(myNetConnection1);
video1.attachVideo(myNetStream1);

myNetStream1.onStatus = function(infoObject:Object) {
    if (infoObject.code == "NetStream.Play.Stop" && infoObject.level == "status") {
        myNetStream1.seek(0);
        myNetStream1.pause(false);
    }
}

// ================================== \\
```

6 Change every instance of the number 1 with the number 2. When you're done, the pasted ActionScript should read like this:

```
// video 2
var myNetConnection2:NetConnection =
new NetConnection();
myNetConnection2.connect(null);
var myNetStream2:NetStream = new Net
Stream(myNetConnection2);
video2.attachVideo(myNetStream2);
myNetStream2.onStatus =
function(infoObject:Object) {
if (infoObject.code == "NetStream.
Play.Stop" && infoObject.level ==
"status") {
myNetStream2.seek(0);
myNetStream2.pause(false);
        }
}
// ================================ \\
```

7 Repeat the process again, but now change each instance of the number with a 3, as above. You haven't actually written any code yet that tells *Flash* which video should be loaded, and which of your NetStream objects each FLV should be assigned to. Next you'll write some ActionScript that will offset each identical video clip by a few hundred milliseconds by loading them at different times. The block of code you write first will tell *Flash* to load one of the FLV files. Then the next time the code is executed it will load the next FLV in sequence. This loop will continue until it reaches the last video.

```
// ================================== \\

function playMovies() {
    if (i <= totalMovies) {
        _root["myNetStream" + i].play("muybridge.flv");
        i ++;
    }
    if (i > totalMovies) {
        clearInterval(playInterval);
    }
}

playInterval = setInterval(this, "playMovies", startDelay);
```

8 Click at the end of the last action in the Actions Panel and create two empty line breaks. Then, type the following:

```
function playMovies() {
        if (i <= totalMovies) {
_root["myNetStream" +
i].play("muybridge.flv");
                i ++;
        }
        if (i > totalMovies) {
        clearInterval(playInterval);
        }
}
playInterval = setInterval(this,
"playMovies", startDelay);
```

This says, "Every so often, if there is a movie to load, load it and attach it to one of the NetStream objects (that you created earlier). If there are no movies to load, stop asking."

9 Scroll to the top of the Actions panel, and click before the top-most action. Create two empty line breaks, click at the top-most empty line break and type:

```
// instantiate some variables
var i:Number = 1;
// define total number of movies
var totalMovies:Number = 3;
// define the start delay between movies (in milliseconds)
var startDelay:Number = 200;

// video 1

var myNetConnection1:NetConnection = new NetConnection();
myNetConnection1.connect(null);

var myNetStream1:NetStream = new NetStream(myNetConnection1);
video1.attachVideo(myNetStream1);
```

```
// instantiate some variables
var i:Number = 1;
// define total number of movies
var totalMovies:Number = 3;
// define the start delay between movies (in milliseconds)
var startDelay:Number = 200;
```

Here, 'i' defines a starting number for the loading sequence and totalMovies is a variable that stores how many movies there are to load. startDelay defines, in milliseconds, the time delay between each movie loading.

10 You can now test your movie by choosing Control→TestMovie. You should see the zoetrope animation play and the videos load in and loop continuously. The movies should appear slightly offset in time, as should the optical illusion created by an actual zoetrope. arts

AN ALTERNATE METHOD

An alternate, and possibly better, way of telling each clip to delay slightly before starting each consecutive video would be to use the 'seek' method of the NetStream class to start each clip at a different time. However, because of the short duration of this running horse sequence, you'll find that the technique you employed in step 8 works better. You can read more about the 'seek' method of the NetStream class by opening the Help panel and choosing Action Script2.0Language Reference→Action ScriptClasses→Net Stream→Seek(Net Stream.seek method).