

10 Loading and Optimizing Flash Content

Lesson overview

In this lesson, you'll learn how to do the following:

- *Load a SWF file*
- Monitor the loading progress
- Use ActionScript to animate a preloader
- Work with dynamic text
- Make symbols visible and invisible
- Cache bitmaps to improve performance

This lesson will take less than an hour to complete. If needed, remove the previous lesson folder from your hard drive, and copy the Lesson10 folder onto it.

Getting started

You'll start the lesson by viewing the finished movie, which includes a preloader similar to the one you created in Lesson 2.

1 Double-click the 10End.swf file in the Lesson10/10End folder to view the final movie.

The preloader—a glass filling with fizzy water—appears. As the promotional movie for the Aqua Zero fizzy drink company loads, the glass fills and the percentage printed beneath the glass increases. When the entire movie has loaded, the preloader disappears and the movie plays.

The preloader and promotional movie have already been created. In this lesson, you'll use ActionScript to load the movie and to run the preloader until it has loaded.

- 2 Open the 10End.fla file in the Lesson10/10End folder. You may want to compare your working file with it, especially when you're adding ActionScript.
- 3 Open the 10Start.fla file in the Lesson10/10Start folder.



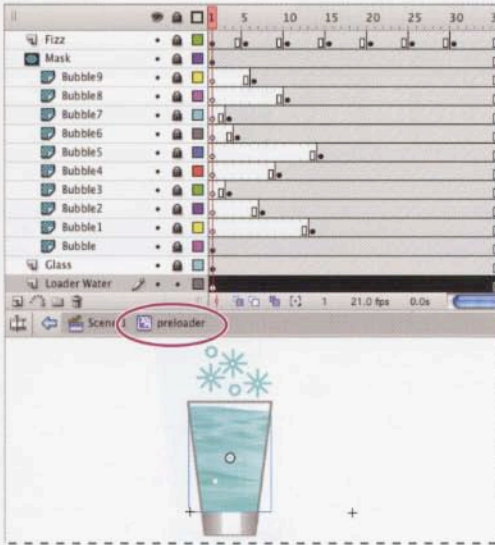
The preloader animation is already on the Stage.

- 4 Choose File > Save As. Name the file **10_workingcopy.fla**, and save it in the 10Start folder. Saving a working copy ensures that the original start file will be available if you wish to start over.

Assembling the preloader elements

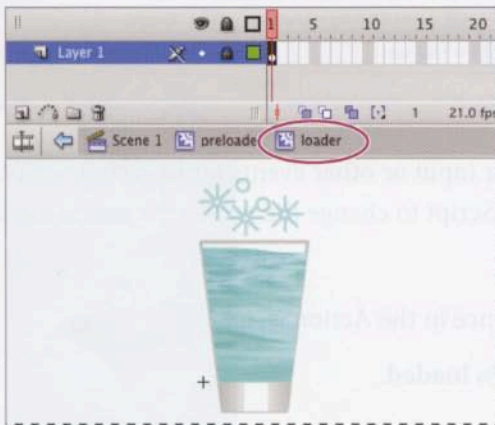
When you first open the 10Start.fla file, the bubbles in the water are animated, but the glass is always full. You'll add a mask that you can animate with ActionScript, so that it reveals the liquid in the glass as the movie loads. You'll also add a text box that displays the percentage of the movie that has loaded.

- 1 Double-click the glass on the Stage to edit the preloader symbol.



The preloader symbol contains several layers that include the drawn elements and that animate the bubbles and fizz. When you select the liquid in the glass object, the Property inspector reports that its instance name is `loadingBar_mc`.

- 2 Double-click the glass on Stage again to edit the loader symbol.



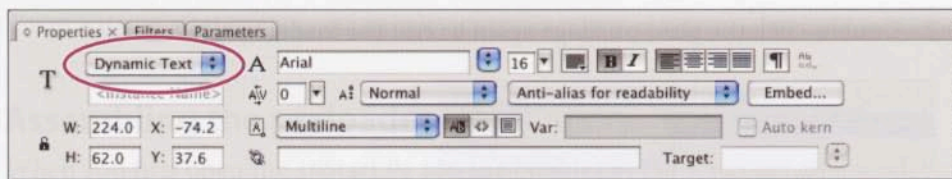
The loader symbol contains only one layer, which contains the water shape. You'll add text and a mask to hide the water.

- 3 Click the Insert Layer icon, and name the new layer **Text**. Rename Layer 1 **Liquid**.

- 4 Select the first frame in the Text layer.
- 5 With the Text tool, drag a text box beneath the glass.



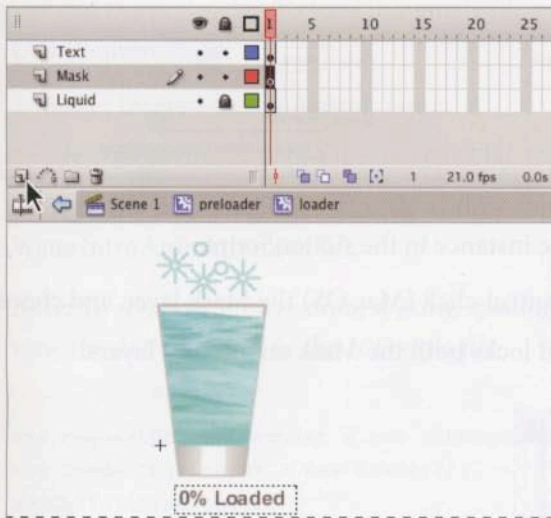
- 6 In the Property inspector, select Dynamic Text from the menu above the instance name. Select Arial for the typeface, 16 for the text size, and a dark gray (#666666) for the text color.



Dynamic text changes based on viewer input or other events in the ActionScript. For this lesson, you will be writing ActionScript to change the text as the movie loads.

- 7 Name the instance **loaderText_txt**.
You'll refer to the loaderText_txt instance in the ActionScript.
- 8 In the text box on the Stage, type **0% loaded**.

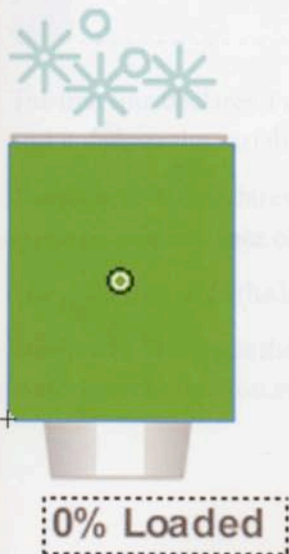
9 Use the Selection tool to resize and reposition the text box so that it is centered below the glass.



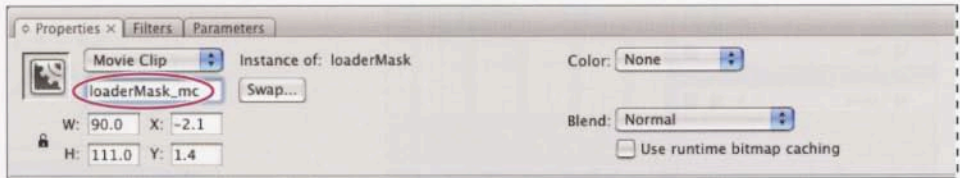
10 Select the *Liquid* layer, and click the *Insert Layer* icon. Name the new layer *Mask*.

11 Select frame 1 in the *Mask* layer.

12 Drag the loaderMask symbol from the Library panel onto the Stage. Position it so that it covers the liquid in the glass.

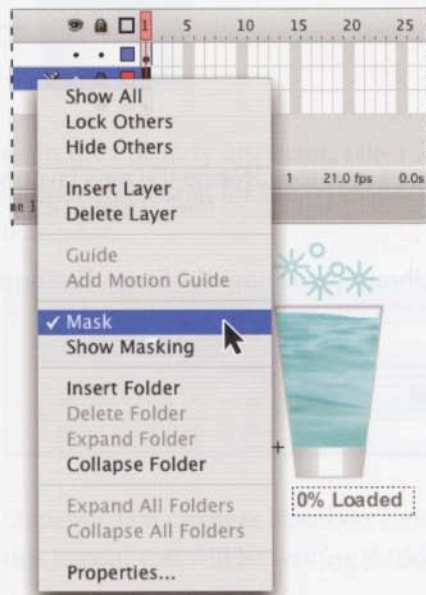


13 Select the instance of the loaderMask symbol on the Stage. In the Properties inspector, name the instance **loaderMask_mc**.



You'll refer to the loaderMask_mc instance in the ActionScript.

14 Right-click (Windows) or Control-click (Mac OS) the Mask layer, and choose Mask. Flash indents the Liquid layer and locks both the Mask and Liquid layers.



15 Click Scene 1 to return to the main Timeline.

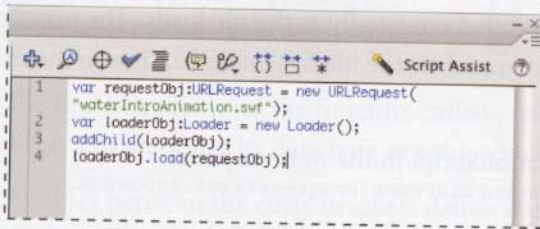
Loading the movie

You'll use ActionScript to load the promotional movie called `WaterIntroAnimation.swf` and the preloader. To load the movie file you'll use a request object called a `URLRequest`.

- 1 Select frame 1 in the Actions layer.
- 2 Press F9 (Windows) or Option+F9 (Mac OS) to open the Actions panel.
- 3 Type the following lines, exactly as they appear here. These lines load the `WaterIntroAnimation.swf` file.

Note: To compare punctuation, spacing, spelling, or any other aspects of the ActionScript, view the Actions panel in the `10End.fla` file.

```
var requestObj:URLRequest = new URLRequest("waterIntroAnimation.swf");  
var loaderObj:Loader = new Loader();  
addChild(loaderObj);  
loaderObj.load(requestObj);
```



The first line declares a variable called `requestObj`, which is of the type `URLRequest`, and it defines the variable as a new request for the file called `WaterIntroAnimation.swf`.

The second line declares a variable called `loaderObj`, which is of the type `Loader`, and it creates a new instance of the class `Loader`.

The third line adds the `loaderObj` variable as a display object that is added to the Stage.

The fourth line loads the `requestObj` variable, which was defined in the first line as the `WaterIntroAnimation.swf` file.

- 4 Choose Control > Test Movie to see your movie so far.

The preloader appears as the movie loads, but the preloader doesn't change. You'll add more ActionScript to animate the text and the mask to reflect how much of the movie has loaded.



To see how your movie will appear when it's downloading, choose View > Simulate Download Settings in the preview window. To change the simulation for different speeds, such as a 56K modem, DSL, or a T1 line, choose View > Download Settings.

Animating the preloader

You used event listeners in earlier lessons. An event listener waits for an event, such as a mouse click, and then reports the event so that the appropriate response can occur. In this case, you want to monitor how much of the SWF file has loaded and display that information on the screen. That is, you're looking for progress.

You'll add an event listener to register and report the progress of the preloader. Then, you'll define a function that will be called repeatedly as the file loads. The function will update the text box that reports how much of the file has loaded and raise the mask to reveal more of the liquid.

- 1 Add the following line to the ActionScript in the Actions panel:

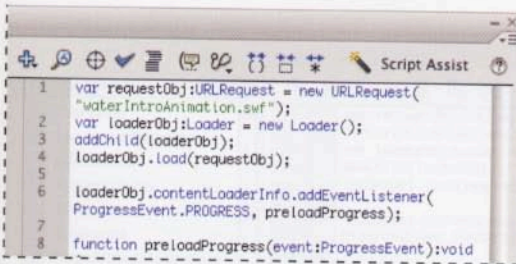
```
loaderObj.contentLoaderInfo.addEventListener(ProgressEvent.PROGRESS,  
preloadProgress);
```

```
1 var requestObj:URLRequest = new URLRequest(  
  "waterIntroAnimation.swf");  
2 var loaderObj:Loader = new Loader();  
3 addChild(loaderObj);  
4 loaderObj.load(requestObj);  
5  
6 loaderObj.contentLoaderInfo.addEventListener(  
  ProgressEvent.PROGRESS, preloadProgress);
```

This line adds an event listener of the `ProgressEvent` type that listens to the `loaderObj` variable you defined earlier. Once the file has started loading, ActionScript creates a `LoaderInfo` object. You're using the `contentLoaderInfo` property to access the information in that object. The event listener, which is also a function, is called *preloadProgress*.

2 Add the following lines to define the function called *preloadProgress*

```
function preloadProgress(event:ProgressEvent):void {  
    var loadedPercent:int = event.bytesLoaded /event.bytesTotal * 100;  
    preloader_mc.loadingBar_mc.loaderText_txt.text =  
loadedPercent + "%";  
    preloader_mc.loadingBar_mc.loaderMask_mc.scaleY = loadedPercent / 100;  
}
```



The first line identifies the function called *preloadProgress* as a `ProgressEvent` type of event, and identifies its data type as `void`, which means that it is undefined.

The lines between the curly brackets define the function itself. First, you've declared a variable called *loadedPercent*; `int` indicates that it is an integer; its value equals the result of the bytes that have been loaded divided by the total bytes times 100.

The next line provides the path to the dynamic text. You named the instance `loaderText_txt`, and it's nested inside the `loadingBar_mc` instance, which is in turn nested inside the `preloader_mc` instance. This line calls for text equal to the value of the `loadedPercent` variable plus the `%` sign, which is included in a string.

Note: You could put any text in the string. For example, you could add the word "loaded" or precede the `loadedPercent` variable with the words, "Please don't leave. You're at" so that it would appear on the screen as "Please don't leave. You're at 14%".

The final line in the function traces the path to the mask, which you named `loaderMask_mc`. Like the text instance, it is nested inside the `loadingBar_mc` instance, which is nested inside the `preloader_mc` instance. This line scales the mask on its y axis (the vertical axis) to equal the value of the `loadedPercent` variable divided by 100.

All functions are surrounded by curly brackets, so a closing curly bracket completes this function.

3 Choose **Control > Test Movie** to preview the movie. In the preview window, choose **View > Simulate Download** to see the animated preloader.



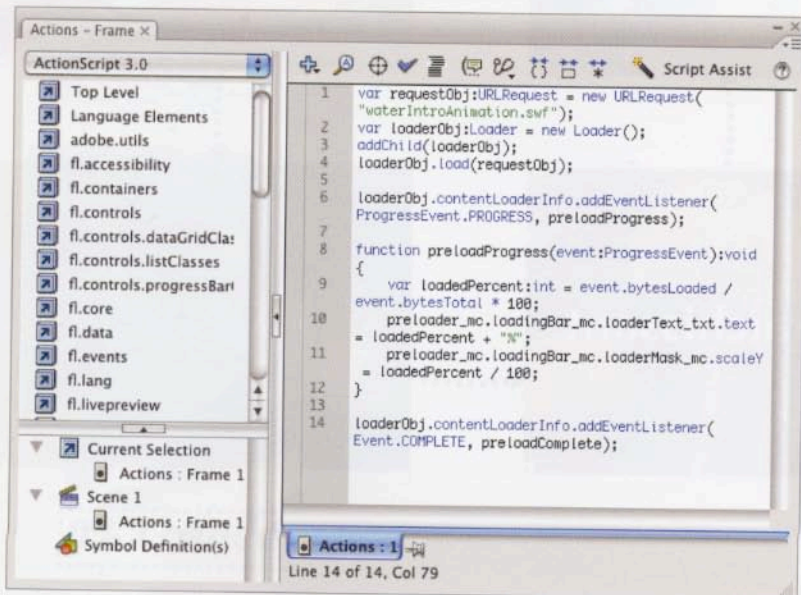
As the movie loads, the glass fills and the percentage changes on the screen.

Changing the visibility of movie clips

The preloader does its job now: you can tell how much of the movie is loaded, and it's entertaining. However, if you watch the movie long enough, you'll see that the preloader appears whenever the movie fades. To correct that problem, you'll change the visibility of the preloader instance so that it no longer appears once the movie has loaded.

1 In the Actions panel, add an event listener to determine when the file has loaded:
`loaderObj.contentLoaderInfo.addEventListener(Event.COMPLETE, preloadComplete);`

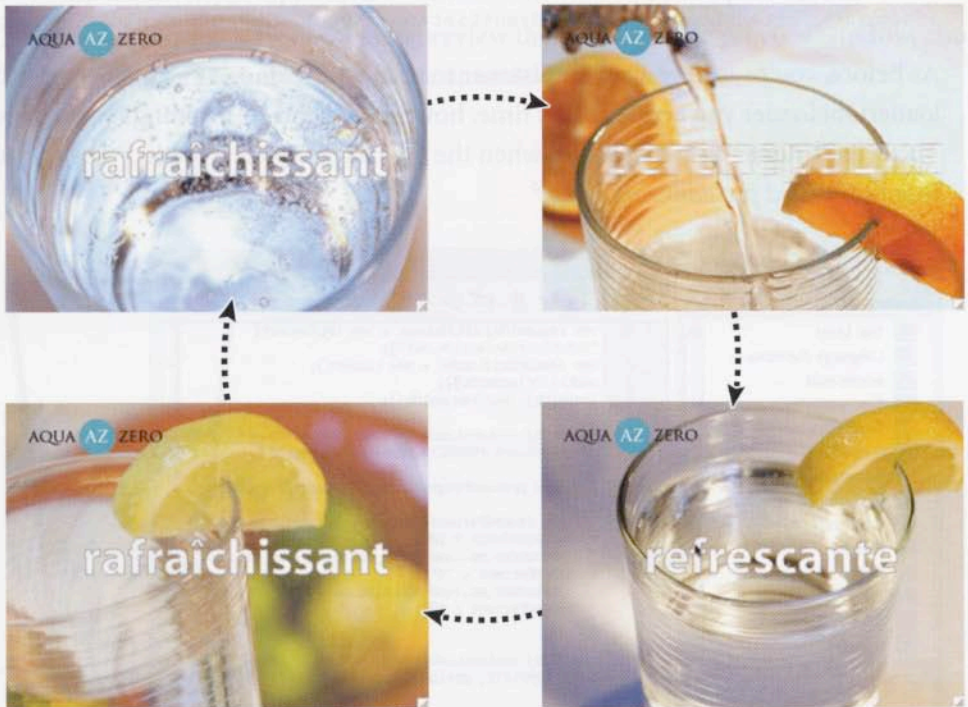
As before, you're adding an event listener to access the `LoaderInfo` object for the `loaderObj` loader you created. This time, however, you aren't listening for progress. The event listener only dispatches when the file has completely loaded. Then it calls the function called *preloadComplete*.



2 Define the `preloadComplete` function by adding these lines in the Actions panel:


```
function preloadComplete(event:Event):void {  
    preloader_mc.visible = false;  
}
```

In this function, which is called when the movie has loaded, the visible property of the `preloader_mc` instance is set to *false*, meaning that it is no longer visible.

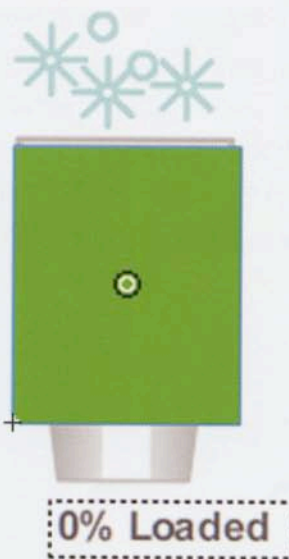


Caching bitmaps

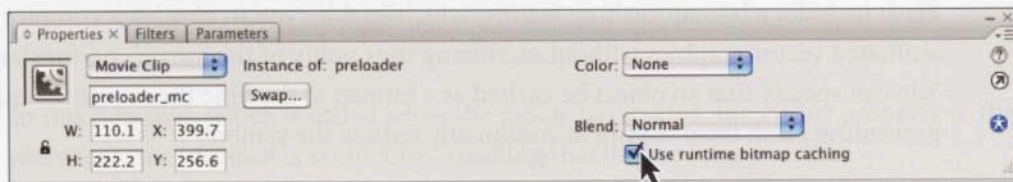
Flash includes a bitmap caching feature, which enables you to play back complex animated vector graphics without sacrificing the quality of the movie's performance. You can specify that an object be cached as a bitmap at the time the movie is run, preventing Flash from having to continually redraw the symbol.

 *Bitmap caching is ideal for complex movie clips where the position of the objects changes, but the shape or content does not change.*

- 1 Close the Actions panel.
- 2 Select the Preloader layer, and then select the preloader_mc instance on the Stage.



3 In the Property inspector, select Use Runtime Bitmap Caching.



Note: You can apply bitmap caching in the Property inspector, or you can simply add a line of ActionScript. To apply bitmap caching through ActionScript for this project, you would add this line of script:

```
preloader_mc.cacheAsBitmap = true;
```



The fizzy drink lets the viewer know that the main site is loading, and keeps the viewer entertained at the same time. You can use this basic process to activate a preloader for a wide range of websites.