**Creating animations**

**Step 1: Set up your project**

During this project, you'll create two ActionScript 3.0 .fla files in Flash Professional. The first file contains the animation that creates the sprite sheet, and the second file contains the code that implements it. In this step, you'll set up your project folder and create a new Flash file.

Before you get started:

1. Download the [project files](http://download.macromedia.com/pub/inspire/1212/flcs6_project.zip) (ZIP, 273 KB) and unzip the archive. Save your published files in this folder and use the supplied assets to follow the steps in the tutorial.
2. Download the [completed project files](http://download.macromedia.com/pub/inspire/1212/flcs6_project_completed.zip) (ZIP, 959 KB) to use as a reference. You can deconstruct these files if you run into any problems.
3. Download the [free trial of Flash Professional CS6](http://www.adobe.com/go/tryflash/) and install the application if you haven't already done so. You can also use your [Creative Cloud membership](http://www.adobe.com/go/creativecloud/) to download and install the product.

Create a new file and start the project:

1. Open Adobe Flash Professional CS6 and choose File > New to create a new ActionScript 3.0 .fla file.
2. Save the file to the project folder as **animation.fla**.
3. Take a moment to explore the Flash workspace.

The Flash workspace is composed of a stage where you see your graphics, a Tools panel, a Timeline panel containing layers and frames, a Library panel where reusable assets are stored, and a handful of other panels used to manage color, transformations, and more (see Figure 2).



Figure 2. The Flash Professional CS6 workspace.

**Step 2: Create an animation in a movie clip symbol**

The first step in creating a sprite sheet is to create an animation inside of a movie clip symbol. Symbols are reusable objects stored in the Library panel. They contain their own timeline, layers, and animation area on the canvas. In this step, you'll create a movie clip symbol and add an animation to it.

Create an animation inside a symbol:

1. Open the Library panel in Flash.
2. Choose Insert > New Symbol or click the New Symbol button at the bottom of the Library panel.
3. In the Create New Symbol dialog box, type the name **walkcycle** and click OK. Notice that the stage changes because you're now in the symbol editing area.
4. At this point, you could start drawing your own animation frame by frame using keyframes along the frames of the timeline. Or you could build tween animations across the frames. For the purpose of this article, you'll copy and paste a fully developed animation into the timeline.
5. Open the supplied\_animation.fla file in the Assets folder in the supplied files. Notice that the walkcycle for the spaceman character consists of 2 layers and 16 frames on the main timeline. You'll need to copy the frames and paste them into a symbol before you can create the sprite sheet.
6. Select all 16 frames across both layers and right-click the selection. Choose Copy Frames.
7. Return to the walkcycle symbol timeline in the animation.fla file. Right-click the empty **keyframe** on Frame 1 and choose Paste Frames. Notice that the frames of the supplied animation appear on the symbol's timeline (see Figure 3).



Figure 3. The timeline of the walkcycle symbol.

1. At this point, you may want to scale the animation to a smaller size to be efficient or change the location of the registration point, which is currently located in the center of the animation. You can easily make changes to all the frames at once using the Onion Skin and Edit Multiple Frames buttons on the bottom of the Timeline panel (see Figure 4).
2. Save the file.



Figure 4. All 16 frames of the animation are selected using the onion skin controls.

**Step 3: Export the animation as a sprite sheet**

You can select one or more movie clip symbols in the Library panel or on the timeline and launch the Generate Sprite Sheet dialog box. The Generate Sprite Sheet dialog box enables you to set a handful of options for the sprite sheet, including its size, background color, and accompanying data format. In this step, you'll export the sprite sheet setup for use in ActionScript.

Launch the Generate Sprite Sheet dialog box:

1. Right-click the walkcycle symbol in the Library panel and choose Generate Sprite Sheet.
2. In the Generate Sprite Sheet dialog box, use the settings shown in Figure 5.



Figure 5. The Generate Sprite Sheet dialog box.

1. Click Export to export the files.
2. Take a look at the files that were generated in the project folder. If you open the animation.xml file in a text editor, you can see the data used to define the animation frames. If you open animation.png in an image editor, you can see that the animation has been flattened into a single image (see Figure 6).



Figure 6. The animation.png sprite sheet exported from Flash.

**Step 4: Set up your .fla file for the Starling framework**

Now that you have a sprite sheet set up for ActionScript, you need to use ActionScript to load the image and the .xml file and play the animation. I've included a few prebuilt scripts to help make the process easy to set up. All you have to do is link the code to a .fla file and learn a few simple commands.

In addition to the supplied ActionScript code, the scripts use the Starling framework to render the sprite sheet animation and draw the animation to the GPU using Stage3D. The Starling framework is an open-source ActionScript 3.0 2D framework designed for game developers. It provides a lightweight solution for manipulating the Stage3D programming APIs in an intuitive way. If you're curious, you can open the scripts in the Code folder in Flash and take a look under the hood.

In this step, you'll create a new .fla file and link it to the ActionScript resources. Create a new .fla file and assign a document class to it:

1. Create a new ActionScript 3.0 .fla file, name it **spritesheet.fla**, and save it to the project folder. It's important that you save the file next to the Code and Libs folders because you'll be referencing the files inside those folders.
2. Open the Properties panel to see the Stage properties. Enter **code.SpriteSheetStage** in the Class field (see Figure 7). This step links the .fla file to the SpriteSheetStage script as if the script were written on Frame 1 of the main timeline.
3. Save the file.



Figure 7. The stage properties with a document class assigned to the .fla file.

A .swc file is a compiled library of ActionScript code and assets. Here you'll link to the Starling framework .swc file to add the sprite sheet rendering engine.

Link the .fla file to a .swc library:

1. Choose File > ActionScript Settings to launch the Advanced ActionScript 3.0 Settings dialog box.
2. Click the Library Path tab, and click the Plus (+) button to add a new path. Type **./libs/starling.swc** in the field (see Figure 8).
3. Click OK to close the dialog box and save the file.



Figure 8. The Advanced ActionScript 3.0 Settings dialog box.

**Step 5: Add the sprite to your Flash movie**

Now that you've added the document class and linked to the .swc library, the .fla file is ready to render your sprite sheet. From here, you'll be controlling the animation by typing code commands in the Actions panel. It's an easy process. You'll follow a copy and paste workflow.

Whether you're using code to control animations or build an application, a primary concept is event timing. In the case of the sprite sheet, you'll need to set up an event handler function that executes when the Starling framework has initialized and is ready to animate. All code needs to execute from that timing.

In this step, you'll add an event handler and instantiate the animation. Add ActionScript to the Actions panel:

1. Rename Layer 1 on the main timeline of the spritesheet.fla file. Change the name to **Actions** for good measure.
2. Select the keyframe on Frame 1 of the sprite sheet .fla file and choose Window > Actions to open the Actions panel.
3. Copy and paste the following code into the text editor:

import code.SpriteSheetEvent;

import code.SpriteSheetInstance;

var spriteAnimation:SpriteSheetInstance;

function startUp( event:SpriteSheetEvent ):void

{

 spriteAnimation = addSpriteSheet("anim", "animation.xml", "animation.png", 175, 75);

}

addEventListener(SpriteSheetEvent.READY, startUp);

1. Choose Control > Test Movie to preview the file. You should see the animation playing back in the center of the screen.
2. Save the file.

The code you just added did a few things. It imported the scripts you wanted to reference. You can now reference them directly by name. The script added an event handler function and assigned it to the READY event. The function does not execute until the event occurs, ensuring that the Starling framework is ready to render animations. In addition, the script instantiated the animation by calling the addSpriteSheet command. When addSpriteSheet is called, you pass in an instance name for the animation, the path to the sprite sheet .xml file, the path to the sprite sheet .png file, and the *x* and *y*coordinates for where the animation should appear.

See spritesheet-step5.fla in the completed files for a working sample.

**Step 6: Create multiple instances**

If you want to create multiple copies of the same sprite sheet, you can take advantage of the cached .xml and .png files by using the duplicateSpriteSheet command. The duplicateSpriteSheet command creates a new animation instance without loading the sprite sheet assets again. You should always use the duplicateSpriteSheet command after first loading the sprite sheet with the addSpriteSheet command.

In this step, you'll add a few lines of code to the startup function. Create multiple animation instances randomly:

1. Return to the script in the Actions panel and add the following code below the addSpriteSheet line in the startup function:

for(var i:uint=0; i < 8; i++)

{

 var xpos:int = Math.floor(Math.random()\*stage.stageWidth-100);

 var ypos:int = 75;

 duplicateSpriteSheet("anim", " anim"+i, xpos, ypos);

}

1. Choose Control > Test Movie to preview the file. This time you should see multiple copies of the animation placed randomly horizontally across the screen (see Figure 9).
2. Save the file.



Figure 9. The walkcycle animation duplicated across the screen.

The code you just added fires up a loop and loops eight times. Each time the loop chooses a random *x* coordinate and calls the duplicateSpriteSheet function to create a new animation instance. The duplicateSpriteSheet command expects you to pass it the name of the instance animation you want to duplicate, a new name for the new instance, and the *x* and*y* coordinates where the animation should appear.

See spritesheet-step6.fla in the completed files for a working sample.

**Step 7: Tween the sprite sheet instance**

You might want to animate the sprite sheet with a tween. For example, because the walkcycle animation walks in place, you'll want to move it in the intended direction using a tween animation. This is totally possible, but you may run into a timing issue if the sprite sheet .png file has not loaded yet. To get around this, you can listen to the sprite sheet instance's SPRITE\_SHEET\_INIT event and then animate the sprite sheet's movieClip property. Waiting for the event is only necessary when using addSpriteSheet.

Before you get started, note that the movieClip property is not a normal Flash movie clip; it's a Starling movie clip. Starling uses hardware acceleration to render the animation in the GPU as opposed to Flash Player. The animation is not in the Flash display list. Instead it exists in a layer below any Flash display object. In this step, you'll animate the walkcycle by tweening it.

Use the [Tween class](http://help.adobe.com/en_US/FlashPlatform/reference/actionscript/3/fl/transitions/Tween.html) with the sprite sheet:

1. Delete the loop code you added in the last section so that you're looking at a single animation again.
2. Add the following import statements below the other import statement:

import fl.transitions.Tween;

import fl.transitions.easing.None;

This code imports the Tween class.

1. Add the following line of code below the addSpriteSheet line. This line assigns an event handler function to the instance.

spriteAnimation.addEventListener(SpriteSheetEvent.SPRITE\_SHEET\_INIT, onSpriteSheetInit);

1. Add the following event handler code below all other lines in the script. This code waits for the SPRITE\_SHEET\_INIT event before beginning the animation.

function onSpriteSheetInit( event:SpriteSheetEvent ):void

{

 var tween:Tween = new Tween(spriteAnimation.movieClip, "x", None.easeOut, 100, 300, 2, true);

}

1. Preview the file. This time you should see the animation load and tween across the screen.
2. Save the file.

The code you just added created an event handler function that waits for the sprite sheet to be loaded before starting the animation. Then it uses the Tween class to tween the sprite sheet on its *x* axis.

See spritesheet-step7.fla in the completed files for a working sample.

**Step 8: Export the sprite sheet for JavaScript**

You can also place the animation on a web page using JavaScript. First you'll need to export the sprite sheet setup for JavaScript using EaselJS.

The benefit of working with JavaScript and HTML5 is the ability to bring rich animations developed in Flash into a web standard environment. That means that browsers and devices that can view HTML5 don't have to have Flash Player installed to view the content. EaselJS is a JavaScript framework that enables you to design in the HTML5 canvas in a way similar to working with the Flash display list. In this step, you'll export the animation again, but this time you'll set it up for JavaScript.

Export for JavaScript:

1. Return to the animation.fla file and select the animation symbol in the Library panel.
2. Launch the Generate Sprite Sheet dialog box and use the same settings as Figure 5, except this time change the Data Format field to EaselJS (see Figure 10).
3. Click Export to export the files.



Figure 10. Switching the sprite sheet to export to EaselJS for JavaScript.

**Step 9: Add the sprite sheet to an HTML page**

Adding the sprite sheet to a web page is easy. You can link to the EaselJS script in the Libs folder and the animation.js file created in the last step. In this step, you'll set up an HTML page to show the animation.

Import the JavaScript and create an instance:

1. Create a new HTML page in Dreamweaver or another text editor, and save it as **animation.html** in the project folder.
2. Delete any code in the HTML page and paste the following code in its place.

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

 <script>var createjs = window;</script>

 <script type="text/javascript" src="libs/easeljs-0.5.0.min.js"></script>

 <script type="text/javascript" src="animation.js"></script>

 <script>

 var canvas, stage;

 function init(){

 canvas = document.getElementById("testCanvas");

 stage = new Stage(canvas);

 var anim = new walkcycle();

 anim.x = 200;

 anim.y = 200;

 stage.addChild(anim);

 Ticker.setFPS(30);

 Ticker.addListener(stage);

 }

 </script>

</head>

<body onload="init();">

 <div class="canvasHolder">

 <canvas id="testCanvas" width="1000" height="500"></canvas>

 </div>

</body>

</html>

1. View the page in a browser to check your work.

The code you just added to the web page links to the EaselJS script in the Libs folder and the animation.js file. When the onload event fires in the browser, an instance of the animation is created and placed inside the testCanvas div tag. The only thing you'll need to change about this code snippet if you reuse it is the name of the sprite sheet object (the line that reads **var anim = new walkcycle()**). The name of the sprite sheet will always be the same as the name of the symbol you published from Flash (in this case **walkcycle**).

See spritesheet-step9.html in the completed files for a working sample.

**Step 10: Create multiple instances in JavaScript**

You can add multiple instances of the animation in JavaScript the same way you can in ActionScript. In fact, the loop code looks almost exactly the same in JavaScript. In this step, you'll instantiate multiple instances of the sprite sheet in JavaScript.

Create multiple instances on the HTML5 canvas:

1. Replace the code for the init function with the following:

function init(){

 canvas = document.getElementById("testCanvas");

 stage = new Stage(canvas);

 stage.width = canvas.width;

 stage.height = canvas.height;

 for(var i=0; i<10; i++)

 {

 var anim = new walkcycle();

 anim.x = Math.random()\*stage.width;

 anim.y = 150;

 stage.addChild(anim)

 }

 Ticker.setFPS(30);

 Ticker.addListener(stage);

}

1. View the page in a browser to check your work.

You can see that the JavaScript looks a lot like the ActionScript. See spritesheet-step10.html in the completed files for a working sample.

**Vocabulary:**

edit bar – panes so scénami

stage – pracovná plocha

docked panels – ovládacie panely

properties – možnosti

tools – nástroje

actions – skripty

timeline/layers – časová os/vrstvy

library – knižnica súborov

keyframe – kľúčový snímok