Dynamic Animation with Fuse Kit – Free!

At some point, most Flash developers get a project where you have to do motion animation in code instead of tweens in the timeline. Video games, simulations or custom UI components are obvious places where code based animation may become necessary. But animating with code can require a lot of lines of code.

The Tween class

When Macromedia made the v2 UI Components in Flash MX 2004, they included a class used in the component infrastructure called "Tween". The **Tween class** allows you to change a property (such as **_x**, **_y**, **_rotation**, **_alpha**, etc.) of a **MovieClip** over time, with various built in easing effects.

There are a few problems with using the Tween class. The first is that a single instance of a Tween class can only animate one property of a MovieClip. So, if you want to adjust _x, _y, _rotation and _alpha, you need four Tween instances (one for each property). The second draw back of Tween, is that there are a large number of parameters needed to create an instance, so it's a long line of code. This compounds the first problem of needing to create multiple Tween instances, but making each instance a painfully long line of code.

The last major drawback with Tween is that if you don't manage it well and clear out completed or unnecessary Tweens, you can end up in a situation where Tweens just keep on animating uncontrollably, similar to having a **setInterval** call run off on you when you don't clear the interval religiously.

Fuse Kit

An open-source alternative to the Tween class called **Fuse Kit**, written by Moses Gunesch, combines a number of previous people's work into a single and very powerful ActionScript Animation kit. It's a kit because you can choose which assets you want to use and only those assets will be compiled into the SWF. So if you are only going to use limited assets in the Fuse Kit, you can choose to create very light SWFs. But if you are going to use some of the higher-end features, you'll end up with a larger SWF file, but you'll have some great code based animations.

We'll take a look at each of the classes that make up Fuse Kit and how you can use them to create some very robust code driven animation.

Installing Fuse Kit

To start with, you'll need to download the latest version of Fuse Kit (currently version 2.0 as of this writing) at http://www.mosessupposes.com/Fuse/.

If you unzip the Fuse Kit source package, you'll find a folder called something similar to **fuse1.1z3**. Inside that folder, you'll find the code of the Fuse Kit. Inside this folder, you should see three more folders, "**com**", "**examples**" and "**fuse2.0docs**" and an **MXP file** called **Fuse2.0.mxp**. If you double click to execute the MXP file, **Macromedia's Extension Manager** should open up and install the Zigo prototypes that Fuse Kit uses.

Next, we need to copy the com folder (which contains the ActionScript 2.0 classes that make up Fuse Kit) to a folder where we can make them available to Flash. If you already have a folder set up where you keep class libraies and have a **classpath** set up, copy this com folder to that location and skip past the next section. If you haven't set up a common library folder, we're going to do that now.

Common Library Folder and Classpath

It's good to have a place where you can save ActionScript class files in a single place and use them in all of your projects as needed without having to copy the whole class structure into the working folder of any particular project. You can make this folder anywhere you like (your **Desktop**, right in the root of your C

drive, etc.). I keep mine in a subfolder of **My Documents**. My ActionScript 2.0 classes are kept in MyDocuments\Flash 8\ActionScript. Of you want to create a similar file structure, copy the com folder and all of its contents to this location.

Now, we need to set this location as a classpath in Flash. Adding a class path just tells the Flash compiler to look in this location for imported classes when compiling any SWF. This way you can import Fuse Kit classes at will with FLAs located anywhere on your computer.

To set the classpath, open Flash. Now, go to **Edit > Preferences... > ActionScript** from the drop down menu. Then click on the button that says **ActionScript 2.0 Settings...**. In the window that will open up, you can click on the plus (**+**) button to add a new classpath, you can then click on the button that looks like a gun site with cross hairs to browse to the folder you created. You'll want to add the folder that contains the com foler. Be sure the class path does not actually include the com folder itself, but rather the folder containing the com folder.

First Look at Fuse Kit

Let's get started with a quick example of how easy animating with Fuse Kit is.

As I mentioned early, Fuse Kit is made up of a number of different classes with specific uses. The three main ones we're going to look at are the **ZigoEngine**, **Fuse** and **FuseFMP**.

ZigoEngine

The ZigoEngine goes far back before Fuse Kit. Fuse Kit is actually new next version of the ZigoEngine. Prior to Fuse Kit, you could use the ZigoEngine to alter the pototype of MovieClips to include some additional methods specific to animation (**slideTo**, **alphaTo**, etc.).

I'm not going to get too into the prototype methods, but because Fuse Kit builds off of the ZigoEngine, it is still an option to create animations using these prototype methods alone. For example, my source code package, Example01.fla contains a single MovieClip called Ball and the following ActionScript code on the root timeline.

```
import com.mosesSupposes.fuse.*;
ZigoEngine.simpleSetup(Shortcuts);
ball.slideTo(Stage.width/2, Stage.height/2, 1);
```

<embed example01.swf>

These few lines of code imports the entire fuse package (including the ZigoEngine). It sets up the ZigoEngine to extend the MovieClip prototype to include the hold animation shortcuts. We can then use the new slideTo method of the ball MovieClip to slide it to the center of the stage from it's current position in one second.

The ZigoEngine shortcuts are well documented in the Fuse Kit documentation, but since we don't really use these when using the rest of Fuse Kit, I'm going to leave that for your discovery and get into examples that show why Fuse Kit is a huge advance over the older ZigoEngine shortcuts.

Fuse

The ZigoEngine prototype shortcuts are kind of the old school ways of using the ZigoEngine. The new school and amazingly powerful way is with Fuse. They describe Fuse as a sequencer. Like an audio sequencer that plays one sound loop, then when that sound is over, it plays the next one in order, kind of like an array of sound loops. Well, Fuse does the same thing with animations. You set up an animation object (which is just a

normal Object litteral that we use all the time in Flash) to describe a particular animation, and then you add it onto the Fuse stack, which is just like pushing an object into an **Array**.

Our second example (Example02.fla) shows how to do the same thing we did in the previous example, but with a Fuse instead of the ZigoEngine shortcuts. Same structure in the FLA, but this time around, our code looks like this:

```
import com.mosesSupposes.fuse.*;
ZigoEngine.register(Fuse);
var f:Fuse = new Fuse();
f.push({target:ball, x:Stage.width/2, y:Stage.height/2, seconds:1});
f.start();
```

<embed Example02.swf>

You may be saying, "but this does the same thing and it's almost twice the lines of code!" And, you'd be correct. But this is just the beginning. We're going to walk with Fuse before we run. Let's take a look at the code we have now and see what it does.

The first line imports the fuse classes, just like before.

The second line initializes the ZigoEngine a little differently before. Rather than simpleSetup, we just want to register the Fuse class with the Zigo engine. This tells the ZigoEngine that we're going to be using a Fuse and it also uses the Fuse class so Flash will compile the Fuse class into our swf.

The third line creates an instance of a Fuse.

Now that we have a Fuse called "f", it works a lot like an Array. It has a push method where we can push an animation object onto the fuse stack. The animation object is just a set of name value pairs that describe a particular animation. In this case, our object declares that the target of the animation is our **ball**. It should move it's _x attribute to the middle of the stage, as is the case with the _y. And it should do the animation over a period of one second.

Finally, we have to call the start method of the Fuse instance. I forget to do this all the time. I forgot to do this when writing the code above. So, if you ever configure your Fuse and nothing happens, the first thing to check is that you remembered to call the start method to light the fuse and get the animation started.

Complex Fuse Animation

The reason for using Fuse instead of the ZigoEngine shortcuts is that you can stack animations into a fuse to create really complex sequences of animations. By pushing multiple animation objects into a fuse before starting it, you can set up huge animation sequences will occur in order based on the order you push them into the Fuse.

For example, let's take the exact same code, but add a few extra animation objects onto the fuse to make the ball walk around the stage (Example03.fla):

```
import com.mosesSupposes.fuse.*;
ZigoEngine.register(Fuse);
var f:Fuse = new Fuse();
f.push({target:ball, x:ball._width/2, y:ball._height/2, seconds:1});
f.push({target:ball, x:Stage.width - (ball._width/2), y:ball._height/2, seconds:1});
f.push({target:ball, x:Stage.width - (ball._width/2), y:Stage.height -
(ball._height/2), seconds:1});
f.push({target:ball, x:ball._width/2, y:Stage.height - (ball._height/2), seconds:1});
```

```
f.push({target:ball, x:ball._width/2, y:ball._height/2, seconds:1});
f.start();
```

<embed Example03.swf>

FuseFMP

FuseFMP is Fuse Filter Management Prototype. It allows you to very easily set and animate Flash 8 filters just as you would animate other properties of a MovieClip. You wouldn't use these all the time, because Flash MovieClip filters can be very resource intensive, but when you need to animate filters in a very simple way in code, FuseFMP is definitely the way to go.

FuseFMP is both a nice way of writing Flash 8 filters and a great way of animating them. Let's start with a simple example of how to set up FuseFMP and using FuseFMP to create a static filter on our ball. Example04.fla has the following code:

```
import com.mosesSupposes.fuse.*;
ZigoEngine.register(Fuse, FuseFMP);
FuseFMP.writeFilter(ball, "Blur", {blurX:20, blurY:20, quality:3});
var f:Fuse = new Fuse();
f.push({target:ball, x:Stage.width/2, y:Stage.height/2, seconds:1});
f.start();
```

<embed Example04.swf>

The code in Example04 is almost identical to Example02 except for lines 2 and 3.

In line 2, we register FuseFMP as well as Fuse with the ZigoEngine. This declares that we're going to use FuseFMP and will cause the FuseFMP resources to be included in our swf.

Line 3 is where we create a **BlurFilter**. This is just like creating a BlurFilter in Flash, but we can create it and apply it in just one line of code with FuseFMP, which makes it a good choice even if we didn't intend to animate the filter.

Complex Animated FuseFMP

We're going to wrap up by animating a filter and building off of Example03. We want our BlurFilter to blur in the upper left and bottom right parts of the animation, but we want the ball to be in full focus in the top right and bottom left of the stage. We're going to add a simple Blur_blurX and Blur_blurY property to our sequence of animation objects and FuseFMP will handle the rest for us.

In Example05.fla, you'll find the following code:

```
import com.mosesSupposes.fuse.*;
ZigoEngine.register(Fuse, FuseFMP);
FuseFMP.writeFilter(ball, "Blur", {blurX:20, blurY:20, quality:3});
var f:Fuse = new Fuse();
f.push({target:ball, x:ball._width/2, y:ball._height/2, seconds:1});
f.push({target:ball, Blur_blurX:0, Blur_blurY:0, x:Stage.width - (ball._width/2),
y:ball._height/2, seconds:1});
f.push({target:ball, Blur_blurX:20, Blur_blurY:20,  x:Stage.width - (ball._width/2),
y:Stage.height - (ball._height/2), seconds:1});
f.push({target:ball, Blur_blurX:0, Blur_blurY:0,  x:ball._width/2, y:Stage.height -
(ball._height/2), seconds:1});
```

```
f.push({target:ball, Blur_blurX:20, Blur_blurY:20,  x:ball._width/2, y:ball._height/2,
seconds:1});
f.start();
```

<embed Example05.swf>

You'll see in the animation objects, we've added Blur_blurX and Blur_blurY properties. FuseFMP handles the tweens for the filter animation based on these values. It automatically calculates the in between blur values and updates the MovieClip filter appropriately.

Conclusion

This article was intended to just wet your apatite for more about Fuse Kit. There are so many more things, including throwing events (by calling functions) when specific animations complete, looping animations, customized easing (see Fuse Kit docs about PennerEasing!!), etc. that I just didn't have room to discuss in this format.

Check out the examples that come with the Fuse Kit package to see how to use it. Hopefully with this introduction, you'll be interested in looking deeper into it. I'd also strongly recommend Lee Brimelow's video demonstrations at his site (http://www.gotoandlearn.com). Lee has some great coverage of Fuse Kit in a very approachable way.

Most importantly, continue to play and experiment with Flash and Fuse Kit. It's really a lot of fun. Once you have a handle on how to use it, the possibilities are endless.