

Making Flash Animations and Incorporating Them in Web Pages

Richard R. Silbar
WhistleSoft, Inc., Los Alamos NM 87544

These days a visit to the World Wide Web usually involves a visual “treat” of things that move, blink, fade in or out, and morph from one shape to another. They sometimes even make noise. Banner advertisements, which are trying to grab your attention, are particularly notorious for using these effects. There are *other* good uses for animation, however, not the least of which is to illustrate a technological or scientific principle [1] that would otherwise be hard to get across in a static, two-dimensional graphic.

There are several ways one can include animations in a web page. Perhaps the oldest is to reference an animated GIF file (GIF89A). JavaScript can produce many animation effects [2]. Java applets [3] and Macromedia Director movies can go even further, creating a high degree of interactivity for the user. In this column, however, I will describe how you can use Macromedia’s Flash software [4] to create and incorporate low-bandwidth animations that allow the web page visitor considerable flexibility in how he or she interacts with the page. In my opinion Flash has by far the easiest learning curve for creating a good-looking animation for the web.

Flash is an inexpensive (\$299) software product that is now in version 4.0. The “shocked” files that it produces are remarkably small, so that embedding a Flash movie in a web page involves only quick downloads. The reason for the small files is that Flash is largely based on vector graphics, rather than bitmaps. The great majority of the web browsers being used today (85% or more) are already capable of playing a Flash movie without forcing the user to retrieve a plug-in from some external web site. That is, the Flash player plug-ins are built-in for Internet Explorer 3.0 and above and Netscape 4.0 and above.

The Flash movie which I am about to describe how to build (Fig. 1) can be viewed at <http://www.whistlesoft.com/~silbar/demo/flash>. You might want to go look at this animation to see where we are going before you delve into the details of how to make it. It involves a well-known equation in physics, $F = ma$, and how to solve it for the acceleration a . To save you an immediate trip to your web browser, however, I’ll describe the animation in words. The “**F**” symbol moves in from the left, an equals sign clunks into place, and the “*m*” and “**a**” symbols move in from the right and stop in the right place. At this point text hints fade in to suggest that, by pointing with the mouse, you can see the definition of each symbol. And, by pressing the mouse when a symbol is highlighted, you see its units as well. Figure 1 shows the animation when the mouse cursor is over the “**F**”. The button below invites you to solve for “**a**”. On pressing the button, the symbols rearrange themselves into “ $a = F/m$ ”, and a new button appears inviting you to start over.

This animation, simple in its physics (but deep), is just an example. You are invited to consider extensions of it that are more appropriate for *your* web pages.

Making the Animation – Part I

I’m assuming you have already installed Flash and have gone through the set of eight short lessons that come in the Help menu. These lessons are quite good tutorials for learning Flash, even if the animations themselves are kind of kitschy (in my opinion). We will create the animation in stages, testing each small increment as we go. As usual, save your work early and often, as you get each piece of the animation working properly.

First, start Flash and save the blank “Movie1” as, say, NewtonEq fla. (FLA is the normal extension for an editable Flash file.) We want the symbols “**F**”, “*m*”, and “**a**” to be active objects so that when the user passes the mouse cursor over them, they pop up a definition of the quantity. Further, if the user then

presses the mouse button, another pop-up will give the symbol's units. Thus we will make the symbols as buttons which respond to the "mouse over" and "mouse down" events.

OK, we'll start with "F". From the Insert pulldown menu, click "New Symbol". Name it "Force" and change its default behavior from Graphic to Button. Clicking "OK" takes you into the editing mode for this button; you'll see four states – Up, Over, Down, and Hit, with the Up state waiting to be defined. We'll want to set each of these states. Go into the type tool (the "A" icon in the toolbox) and type a boldface "F" (24 points is a good font size) at the registration point. Next, click in the Over column, then right-mouse-click to "Insert Keyframe" here. (We need a key frame in the Over column to be able to have something we can edit.) The "F" is still with us, but we'll want its color to turn to red when the mouse cursor passes over it. You can do that by selecting the "F" (still in text mode) and changing to red in the color well present at the bottom of the toolbar. We'll also put the definition pop-up into this mouse state. To do that, drag out a black-outlined box with a white fill above the "F" symbol using the rectangle tool. Now switch back to text mode and type in "Force, a vector" inside the box. It should still be red from the change to the text color well.

Likewise, make the Down State with a blue "F" and a box containing the units ("newtons", in blue) below it. (I chose to keep the red definition appearing also in the down state, but that is up to you.) At this point the "F" button would only react to the mouse when it is exactly over one of the pixels in the "F". This could give your pop-ups the jitters, if your end-user is not steady with the mouse. To cure this, go to the Hit State of the button and drag in a rectangle around the "F". (It won't show up in the actual movie.). This rectangle defines the area around the "F" which responds to the mouse. Save your work.

The "F" button-symbol is now defined, and we will do the same for the other symbols, "m" and "a", in a bit. But first let us test our "F" and see how to animate it. Exit the symbol-editing mode and return to the edit-movie mode by typing <ctl-E>. Double-click on the name "Layer 1" in the time-line to rename it as "Force". Open the library where your "F" symbol-button resides (if it isn't already open) with <ctl-L> and drag an instance of the "F" to the left-hand edge of the stage. It will help you in placing things on the stage if you turn on the Grid and Snap options (in the View Menu). You'll notice that the empty circle in this layer, Frame 1, has filled in, indicating that the initial key frame for this layer now has content.

In the Force layer, click on (say) Frame 15, then right-click and select "Insert Frame". You see that Frames 1 through 15 are now present in the movie, but are still empty. With Frame 15 still selected, right-click and select "Insert Keyframe". Its little circle comes up already filled in, with your "F" present but still at the left-hand edge. Select it and drag it to its desired final position in the equation. To make the left-to-right motion, now click a frame somewhere between 1 and 15. From the right-click menu, select "Create Motion Tween". [The word "tween", derived from "between", stems from perhaps early Walt Disney days and refers to the intermediate frames simulating the motion between two key frames.]

The first animation is ready to run (if all is well). In the Control pulldown menu, select "Rewind" and then "Play". The "F" should move, in about 1 second (at 15 frames per second), from the left edge to its temporarily final resting-place. You can use the VCR-like buttons in the toolbar (or the Enter key) to replay this motion. Normal people reading this column will be motivated to tweak things to improve them. Oh, yes! Now is also a good time to check the mouse-over and mouse-down button actions, to see that the definition and units do pop up properly. If it is working right, save your work. (OK, from here on I'll mostly stop nagging you to save.)

One tweak I felt necessary was to cure the abruptness with which the "F"-motion starts. Let's have the "F" fade in. Select Frame 1 in the Force layer and, from the Modify pulldown menu, click "Instance". In the Instance Properties panel that comes up, choose the "Color Effects" tab. From the list box, select Brightness and move its slider all the way to the right (100%). The preview of the "F" to the left should go blank, that is, fully white. Click OK and test the movie to see that it fades in nicely.

It is now time to start filling in the rest of the equation. Its pieces will all be on separate layers (since we will be animating each symbol separately). So, create a new layer for the equals sign by right-clicking on the Force layer and choosing "Insert Layer". Rename it "Equals" and, to maintain your sense of

mathematical logic, drag it below the Force layer. At this point you might as well also create the “Mass” and “Acceleration” layers also, in the same way. Then, to provide some working space for the new symbols we are about to add, you’ll want extend the movie out to, say, Frame 60 in all these layers. Do this by clicking there, dragging down to select all the layers, and choose “Insert Frame” from the right-click menu. If you test the movie at this point, you will see the “F” coming to a stop and sitting there for another three seconds.

OK, let us put in the equals sign. A few frames after the “F” stop, say Frame 18, make a key frame in the Equals layer. Using the Text tool, type in a “=” (keeping the same font and size as the “F” but not bold). Test. OK, but a little dull, don’t you think? Let’s put in a “clunk” sound at the same time the equals sign appears to help the viewer see it. Insert a new layer, call it “Sounds”, above the Force layer. Open the Sounds “clip art” library, found in the Libraries pulldown menu. Try playing a few until you find one you like. (I liked Brick Drops – I have no idea why they made it plural.) Drag it first into your movie library (remember the <ctl-L> if you need it). Then, after selecting the same frame in the Sound layer as when the equals sign appears, drag the sound from your library into place. As usual, test and enjoy the music.

Now, to bring in the “m” and “a” symbols from the right, we can do pretty much what we did in creating and animating the “F”. The process of creating the new button-symbols is quicker, however, if you just copy the “F” button in the library (using the right-click menu as usual), rename it, and then edit the four button states appropriately. It probably isn’t necessary, of course, to remind you that mass is a scalar and has units of kilograms, while acceleration is a vector with units of meters per second squared. In editing the copies you will probably have to scale the sizes of the pop-up boxes to have their captions fit inside. To get the superscript for “second squared”, bring up the font panel with <ctl-T> and choose “Superscript” in the Position list box. You may also want to tweak the positions of the boxes and texts in the Over and Down States so that the definition doesn’t move when you go from Over to Down. Use the arrow keys for fine positioning.

To make the “m” animation, make a key frame in the Mass layer at Frame 20 and drag in the new “m” symbol from the library, placing it on the right edge of the stage. Insert a new key frame at 35 and drag the “m” into its final position. Select the “m” at Frame 20 and modify this instance’s brightness to 100% for the fade-in. And test. Good? OK, now do the same for the “a” symbol, starting at Frame 30 and ending at 45, again fading in from the right. And test again.

To finish up this half of the animation, make a new symbol named “Hints” and a new layer for it. Have the words shown in Fig. 1 fade in from the “northwest” to their final position. “Hints” need only be a graphic symbol, not a button.

Making the Animation – Part II

We want next to install a button so the viewer can solve this complicated equation for “a”. In the Buttons Library (in the Libraries pulldown menu) select “Push Bar” and drag it into NewtonEq’s library. Rename it “Solve”. It already has most of its properties defined, but it needs a label. Double-click it to go into the editor. Using the Type tool in the Up State, choosing a sans-serif font, type in “Solve for ‘a’” on its face. You may be worried that it will show up on top of a big white box, but you will be relieved to see that box disappear once you deselect the text. Copy the label you just created (select, then <ctl-C>), and paste it into the Over and Down States. Again, you may need to tweak the positions of the label a bit so it doesn’t move when switching states. (On the other hand, you *might* want it to move a bit in the Down State.)

The button is now (almost) ready to use. Create a new layer named “Buttons” and select Frame 56 in it, just after the stopping position of the Hints text. Then drag the “Solve” button from your library into position below the “Hints”. Run it and test. Looks good? Clicking on the button doesn’t *do* anything, you say? Of course not, we haven’t yet told it what to do. Let’s also try another test, that of making a Shockwave file. To do this either click “Test Movie” in the Control pulldown menu or type its keyboard

equivalent, <ctl Enter>. Oops! The movie runs, briefly displays the button, and then starts all over again. You're in an endless loop!

To cure this, create a new layer at the top called "Actions". Insert a key frame in Frame 56 of this layer. This is the same frame as the one where the button appears. Select it, right-click, and choose "Properties". Got to the "Actions" tab and there click on the button with the "+" sign and the little black triangle in the lower right corner. This shows you what actions you can insert at this point. We want "Stop", so click that and OK. Now, run "Test Movie" again with a <ctl-Enter>. (You will probably have to press the Enter key once more after the Shockwave window comes up to get the movie running.) You should see that it does stop when the button appears – no more infinite loop. And you can check that the symbol-buttons are still active in this stopped state.

To give the "Solve" button something to do, first extend the movie by clicking in Frame 120, dragging down to select all layers, and right-click to select "Insert Frame". This will be the area in the time-line for this half of the animation. Now, with the button in Frame 56 selected, click on "Instance" in the Modify menu and go to the Actions tab for this button. From the Add Action list box (accessed from the "+" icon) select "Go To" and set the Frame Number on the right to 61 (instead of the default 1). Also check the "Go to and Play" box to change from the default "Go To and Stop". What shows up in the script box is a Lingo-like function,

```
On (Release)
    Go to and Play (61)
End On
```

that responds to releasing the mouse button after pressing it. Click OK and, as usual, run the animation (hit the Enter key). You'll see the frame pointer move through the button without stopping, going on (without any further changes to the stage) until it stops at Frame 120. You can also test the movie (<ctl-Enter>), in which case the stop at frame 56 will take place. Clicking the "Solve" button then takes you to the end of the movie at Frame 120, but you won't see anything happening at this time. Buttons perform their actions only in the Shockwave movie, not in authoring mode.

To have something happen when the button is pressed, right-click on Frame 61 in the Buttons layer and insert a blank key frame (to erase the button). Then insert (regular) key frames in the Force layer at Frames 61 and 70 and create a motion tween between them. Position the "F" symbol in the second key frame about three grid squares back to the left. This is to make room for the division by "m". Check the motion.

Now create a new graphic symbol in the library, "Divide Sign", which is just the character "/". You want it to be in the library, because you will later animate it when you re-arrange the equation. Make a new layer for it and drag it into position after the "F" in Frame 72 of the Divide Sign layer. There still should be room before the equals sign for the "m", so the equation temporarily looks like "F / m = a".

We want the motion for the "m" to avoid the equals sign by swinging under it. Flash has a way of having a motion follow a guide path, but I find that setting up such a motion is pretty tricky and finicky. A simpler way is to break up the motion tween into pieces with intermediate key frames. Make key frames in the Mass layer, at Frames 75 and 85, and create the motion tween as usual. Add additional key frames at Frames 78 and 82, within the tween. Then using the snap-to-grid, set up the intermediate positions one square down and the final position after the "/". Again, check the motion to see it is what you want.

Now that we have the equation solved for "a", we want to re-arrange it so the "a" is on the left side of the equals sign. This can be done in the same way as you just moved the "m". I chose to move the "a" over the equals sign and the "F", "/", and "m" under it, all simultaneously. I leave the details of doing this to you, based on what you learned above regarding piece-wise tweening. For me, these motions take place between Frames 90 and 100. As a hint, using the key equivalent <F6> to insert a key frame will save you some work.

To finish up this second half of the animation, insert a key frame in the Actions layer at Frame 105 and, as before, set its action to “Stop”. Make a copy of the “Solve” button and change it into a “Start Over” button. Drag this new button into the Buttons layer, also at Frame 105. Double-clicking the button, you get the instance properties panel, where you can set the “On Release” action to “Go To and Play(1)”. If you want, you can select and trim off the excess frames (106 to 120) using “Delete Frame” in the right-click menu. Test for the final time (and save your work).

Figure 2 shows what the timeline for both parts of the animation should look like when all is said and done.

Embedding the Animation in a Web Page

Unfortunately, now comes an ugly part, brought about because the two major web browsers, Netscape and Internet Explorer, do things differently at times. In particular, they deal with Flash Shockwave movies very differently.

The quick way to see the difference also builds an HTML file for you. In Flash’s File pulldown menu, click on “Publish Settings”. Make sure that the Flash and HTML check boxes are checked. Then click the Publish button, and voila!, instant web page. Note that the NewtonEq.swf file is only 12 kilobytes in size.

This is what the Flash-generated HTML looks like (after cleaning it up some and throwing away a lot of unnecessary comments):

```
<HTML>
<HEAD>
<TITLE>NewtonEqA</TITLE>
</HEAD>
<BODY bgcolor="#FFFFFF">
<OBJECT classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
codebase=http://active.macromedia.com/flash2/cabs/swflash.cab#version=4,0,0,0
ID=NewtonEqA WIDTH=550 HEIGHT=400>
  <PARAM NAME=movie VALUE="NewtonEq.swf">
  <PARAM NAME=quality VALUE=high>
  <PARAM NAME=bgcolor VALUE=#FFFFFF>

  <EMBED src="NewtonEq.swf" quality=high bgcolor=#FFFFFF WIDTH=550
HEIGHT=400 TYPE="application/x-shockwave-flash"
PLUGINSOURCE="http://www.macromedia.com/shockwave/download/index.cgi?Pl_Prod_Ver
sion=ShockwaveFlash"></EMBED>

</OBJECT>
</BODY>
</HTML>
```

It is good that Macromedia thought to generate this mess for you pain-free, because I assure you that it is definitely a pain to type in by hand (as I once did). The OBJECT tag is what Internet Explorer uses, and it refers to an ActiveX component that the browser calls upon for displaying the movie. This tag is ignored by Netscape, which uses the EMBED tag instead. If either browser does *not* have the Flash player plug-in, it will then point this out and allow the user to download it (from Macromedia’s site).

This HTML coding *only* shows the NewtonEq movie and nothing else, as you can see for yourself by trying it out in your favorite Browser (off-line). If you want to have the movie amongst other content, and if you want to display some image in place of the movie for those browsers that do *not* have the Flash plug-in installed, then you can use JavaScript to do the checking. Without much further comment, the HTML code would look something like this:

```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HEAD>
<TITLE>WhistleSoft, Inc. </TITLE>
</HEAD>

<BODY>
  <!--Top part of Body Content goes here-->
  <H1> Newton's Second Law</H1>

  <!-- begin OBJECT tag, understood by ActiveX-capable browsers -->
  <OBJECT CLASSID="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
CODEBASE="http://active.macromedia.com/flash2/cabs/swflash.cab#version=2,0,0,0"
WIDTH="550" HEIGHT="400" >
  <PARAM NAME="MOVIE" VALUE="NewtonEq.swf">
  <PARAM NAME="QUALITY" VALUE="high">
  <PARAM NAME="LOOP" VALUE="false">
  <PARAM NAME="PLAY" VALUE="true">

  <SCRIPT LANGUAGE="JavaScript">
    <!-- begin hiding from old browsers
      if (navigator.mimeTypes && navigator.mimeTypes["application/x-shockwave-
flash"]) {
        document.write('<EMBED SRC="./NewtonEq.swf" WIDTH="550" HEIGHT="400"
LOOP="false" PLAY="true"></EMBED>')
      }
      else {
        document.write('<IMG SRC="./images/NewtonEq.gif" WIDTH="329"
HEIGHT="56" ALT="Non-Flash Newton\'s Law" BORDER="0">')
      }
    // done hiding from old browsers -->
  </SCRIPT>
</OBJECT>

<NOSCRIPT>
  <IMG SRC="./images/NewtonEq.gif" WIDTH="329" HEIGHT="56"
  ALT="Non-Flash Newton's Law" BORDER="0">
</NOSCRIPT>

  <!--Rest of Body Content goes here-->
  <P>Neat, isn't it?</P>

</BODY>
</HTML>

```

The Else statement in the JavaScript is for Netscape browsers that do not have an enabled Flash plug-in, and the NOSCRIPT tag handles any other non-Flashing browsers.

Wrapping Up

This column has shown you in fairly complete detail how Macromedia's Flash can be used to produce a rather sophisticated animation of an equation. The resulting Shockwave file is, nonetheless, surprisingly small. As stated earlier, I invite you to consider how you might adapt the present simple example into something that would be appropriate for your web pages.

Force, a vector

$$\mathbf{F} = m \mathbf{a}$$

Move your mouse over a symbol to see its definition.
Then depress the mouse to see its MKS units.

Solve for "a"

Fig. 1. The animation with the mouse cursor over the "F" symbol.

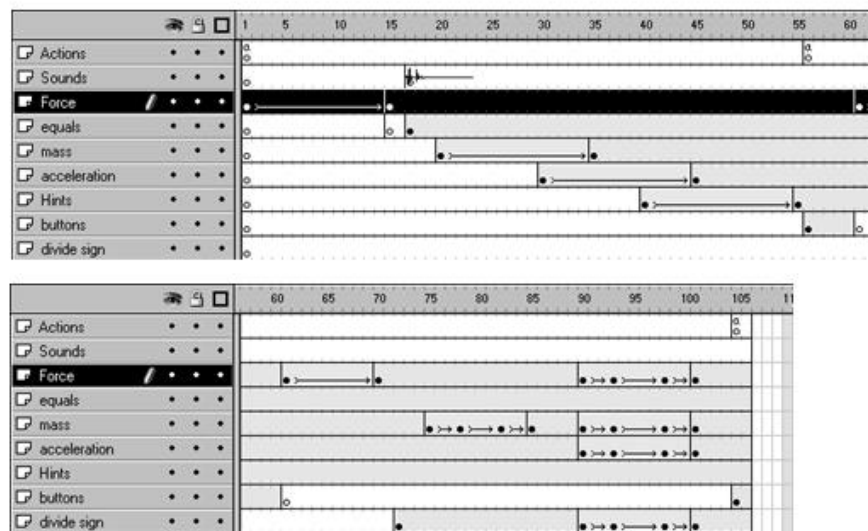


Fig. 2. The timeline for this animation, as seen in the Flash authoring window when the animation is complete. The black dots are at key frame positions and the arrows between them indicate regions of motion tweening.

References

1. R. R. Silbar, W. C. Mead, and R. A. Williams, "Animations in Physics Educational Software," ED-MEDIA '99, Seattle, WA, June 1999 (p. 466 of Conf. Proc., Assoc. for Adv. Of Computing in Education, ISBN 1-880094-35-5).
2. See, e.g., V. Neou and M. Recker, *HTML 3.2 with JavaScript*, Prentice Hall, 1997, ISBN-0-13-270125-1.
3. E.g., W. Christian and A. Titus, "Developing Web-Based Curricula Using Java Applets," *Computers in Physics*, Vol. 12, No. 3, 227, 1998.
4. See <http://www.macromedia.com/products/flash/>. Many good examples of Flash "movies" can be seen and studied at <http://www.shockwave.com>.