I've been working a lot on springs and elasticity and figured I should share what I have learned. Many thanks to Jobe Makar (www.electrotank.com) for introducing me to the subject in Macromedia Flash Super Samurai.

I highly recommend you check out my gravity tutorial, also on this site, as that will give you a lot of the basics, but I'll try to start from close to scratch here anyway.

First of all, let's look at what a spring is. Rather than looking at it as an object, let's look at it as a force. When you start thinking of a spring as a force, it's probably better to use the word "elasticity". After you start looking at it that way, you can start thinking of a lot more ways of using this force than just springs.

So what does this force do? It pulls an object towards a particular point. An important point of this force is that the further away you get from the target point, the stronger the pull. This would be opposed to gravity, where the pull gets weaker with distance. In case you are having any trouble visualizing this, grab a rubber band, hold it against your arm, pull it out an inch or two and let it go. Now pull it out as far as you can and let it snap against your arm. Which force was stronger?

OK, let's start coding. As promised, I'll run through the basics quickly, starting with Actionscript motion.

Make a movie clip, start with a simple ball. Put it on the left hand edge of the stage. Put the following code on the movie clip:

```
onClipEvent(load){
   vx=1;
}
onClipEvent(enterFrame){
   _x+=vx;
}
```

vx is velocity along x (horizontal). You can think of it as speed if you want.

_x+=vx is the same as saying _x=_x+vx. We're just adding the velocity to the x position on each frame. If you test the movie now, you'll see the ball slowly crawl across the stage to the right. OK, if you've gone through the gravity tutorial, you're already way beyond this, so let's move right along. Next we need to know about acceleration. Acceleration can be defined as a CHANGE in speed or velocity. Something is moving slowly, then starts to move faster, or is moving quickly and

slows down. In our Flash file here, this means that not only is _x going to change, but vx is going to change frame by frame. Confused? Let's make it clearer:

We have an acceleration of .1 and a vx of 1.
After the first frame vx will be 1.1, so our ball will be moving slightly faster. Then it will be 1.2, a little faster, then 1.3, 1.4, 1.5, getting faster and faster each frame.

Coding this is pretty easy. Just set the acceleration, we'll call it "ax" and add it to vx each frame, then add vx to _x. It actually looks simpler when you code it:

```
onClipEvent(load){
   vx=1;
   ax=.1;
}
onClipEvent(enterFrame){
   vx+=ax;
   _x+=vx;
}
```

Test the movie now and you'll again see the ball move to the right, but first slowly, and gradually speeding up. Once again, if you've gone through the gravity tutorial, this might look familiar. It is the exact same thing we did with gravity, but on the y axis. That's because gravity, or elasticity, magnetism, or any other force acting to move an object acts as acceleration. It changes the object's speed.

Now, just to show that acceleration can slow down as well as speed up an object, change vx to 5, and change ax to –0.1

The ball should shoot over to the right, but slow down, gradually stop and start moving in the opposite direction. This is the same as gravity acting on an object that is moving in an upward direction. It goes up, slows, stops and starts falling again.

Next step. We've got our _x changing each frame, and our vx changing each frame. Now, believe it or not, we are going to change our ax, acceleration, each frame. As I said before, elasticity is stronger when far from the target point, weaker when closer. And the force is acceleration, so that means we have a stronger or weaker acceleration based on distance. Time to add to our file, so we can see it in action.

First we need to set a target so we have somewhere to spring to. We'll call it "tx" for target x. (Having everything with an x will make it easy to just duplicate everything with a y later on to move to two dimensions.)

Then we will need to find the distance between tx and _x. Our acceleration will be a fraction of that distance. On the next frame, the distance will be smaller, so the acceleration will be smaller. Note that I said the ACCELERATION would be smaller, not the velocity. Our object will continue to speed up, but it won't be speeding up as fast as before.

Eventually, the object is going to hit and move past the target. This is the beautiful part. Because now, the distance is negative. So the acceleration will be negative. In other words, it is going to slow down. And the further it goes, the more it's going to slow down, until it stops and starts moving back – toward the target again. Here we go:

```
onClipEvent(load){
   tx=250;
}
onClipEvent(enterFrame){
   // ax is a fraction of the distance from tx to _x
   ax=(tx-_x)*.2;
   vx+=ax;
   _x+=vx;
}
```

Now you should be able to see your ball wildly bouncing back and forth. But it looks a bit odd because it never slows down. In the real world, there would be friction and loss of energy in various forms which would dampen the motion and acceleration. The easy way to do this is to just decrease the velocity by a small percent. We did this in the gravity tutorial by adding a friction variable. We'll do the same thing here but call it "damp". Also, we're going to move that .2 out of the equation and replace that with a variable called "k". I'm using "k" because that is what it's called in Hooke's law (which is the equation governing elasticity – which is exactly what we are using). "k" is the "springiness" of the spring. Try different values and see what happens. As a note, it's usually good programming practice to remove constant values out of the action part of your code and put them in an initialization section like we are doing here. It makes them easier to find and change as you tweak it to get the exact effect you want. Here we go:

```
onClipEvent(load){
   tx=250;
   k=.2;
```

```
    damp=.9;
}
onClipEvent(enterFrame){
    // ax is a fraction of the distance from tx to _x
    ax=(tx-_x)*k;
    vx+=ax;
    vx*=damp;
    _x+=vx;
}
```

(Once again, vx*=damp is the same as saying vx=vx*damp.)

Now you should have a pretty realistic spring. You can adjust the damp and k variables to change its properties. Of course you are going to want a two dimensional spring eventually, and that is as easy as copying any line with an x in it and changing it to y, basically. Here:

```
onClipEvent(load){
    tx=250;
    ty=250;
    k=.2;
    damp=.9;
}
onClipEvent(enterFrame){
    ax=(tx-_x)*k;
    ay=(ty-_y)*k;
    vx+=ax;
    vy+=ay;
    vx*=damp;
    vy*=damp;
    _x+=vx;
    _y+=vy;
}
```

That should about do it for a two dimensional spring. I'm going to wrap it up with a couple of projects that will demonstrate how you can use it. First, we'll make a draggable ball.

1. Double click on the ball to edit it.
2. Select the whole graphic inside and convert it to a button.
3. On the button put this code:

```
on(press){
   startDrag("");
   drag=true;
}
on(release, releaseOutside){
   stopDrag();
   drag=false;
}
```

This is your basic code for making a draggable clip, with the addition of setting a variable "drag" to true or false depending on whether we are currently dragging or not. We'll need this because we don't want the ball to be springing around while we drag it, just when we let go.

4. To accomplish this, go back to the earlier script and put an if statement in there that covers the whole block of code in the enterFrame section. Now this code will only function when we are not dragging:

```
onClipEvent(enterFrame){
   if(!drag){
      ax=(tx-_x)*k;
      ay=(ty-_y)*k;
      vx+=ax;
      vy+=ay;
      vx*=damp;
      vy*=damp;
      _x+=vx;
      _y+=vy;
   }
}
```

( "if(!drag)" means "if(NOT drag)" or "if drag is NOT true")

Now you can drag the ball around to different points and let it go and watch it spring back to the target. Pretty cool (I think).

Lastly, up to now, we've been using a fixed point as a target: 250, 250. But there is no reason that we can't move that around while the movie plays. Let's make it spring to the mouse cursor. Note

that this project is not really compatible with the last one since they are both using the mouse for different reasons. So go back to the original completed code and in the load section get rid of tx and ty. In the enterFrame section, substitute _root._xmouse and _root._ymouse for tx and tx. That simple. Here it is if you are so lazy you just want to copy and paste:

```
onClipEvent(load){
   k=.2;
   damp=.9;
}
onClipEvent(enterFrame){
   ax=(_root._xmouse -_x)*k;
   ay=(_root._ymouse -_y)*k;
   vx+=ax;
   vy+=ay;
   vx*=damp;
   vy*=damp;
   _x+=vx;
   _y+=vy;
}
```

It still amazes me that such a cool effect can be accomplished in so little code. Now the ball will spring to the mouse, wherever it is. If you don't move the mouse, it will eventually settle down, but otherwise you can swing the ball back and forth, up and down, or around in a circle.

Special bonus: There is no reason why you cannot have more than one force acting on an object. In other words more than one acceleration. Just add them all onto the velocity variables before you add velocity to the position. For example, in the above code, set a "grav" variable equal to 1. Then right after vy+=ay; add vy+=grav;  Now you have elasticity AND gravity working on the same object. Other things to try: have an object springing to two different targets. I'd be interested in any cool files you come up with.