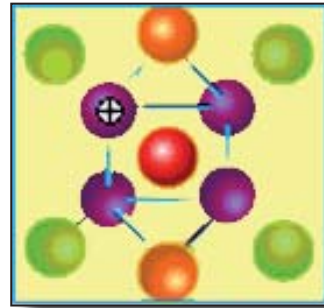


published on 5th September 2006

Generating MovieClips Programmatically



BY JAMES LEE

Chief Instructor

“Not only can you create MovieClip objects at authoring time by dragging instances onto the stage from the library; you can create them programmatically using Actionscript.

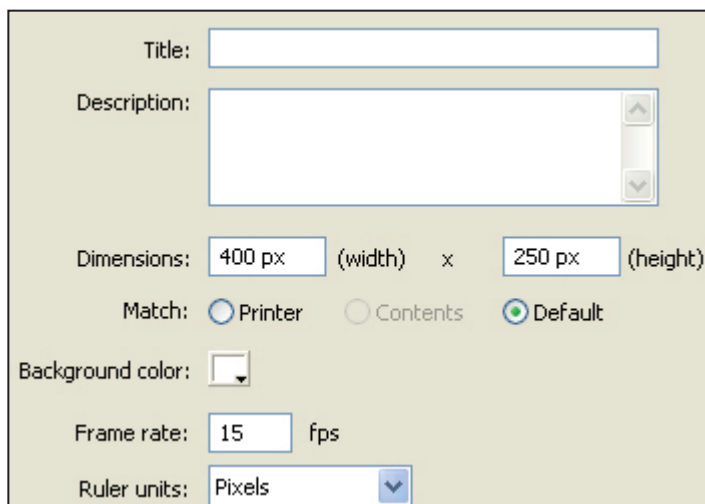
In this tutorial, you will explore some of the techniques for accomplishing this.”



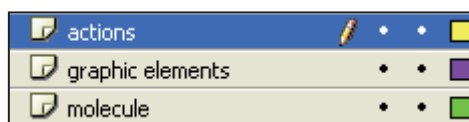
James Lee
*Adobe / Macromedia
Certified Instructor*

Creating assets

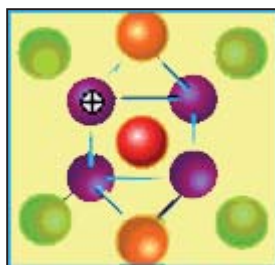
- 1) Create a new file in Flash. Set the size to about **400px** by **250px**.



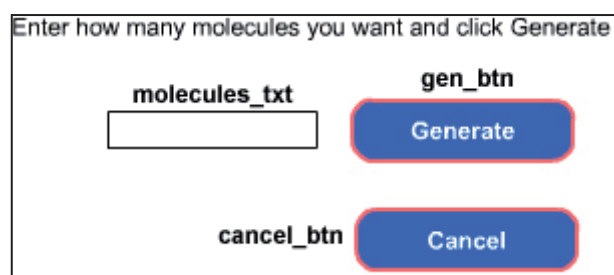
- 2) In Flash, create 3 layers, **molecule**, **graphic elements** and **actions**.



- 3) Within the **molecule** layer, create a nice looking object that looks like a molecule. We are going to create a molecule generator!



- 4) Within the graphic elements layer, create the interface as shown and name it as shown.



Using Loops in Actions

- 5) Go to Actions layer and press **F9** to go to Actions Panel.
- 6) Type in the codes as shown below :

```

1 gen_btn.onRelease = function(){
2 for (var i:Number=0; i<Number(molecules_txt.text); i++) {
3     var scale:Number;
4     molecule_mc.duplicateMovieClip ("mol_mc" + i, i);
5
6     _root["mol_mc"+i]._y = Math.round(Math.random()*200);
7     _root["mol_mc"+i]._x = Math.round(Math.random()*400);
8     scale = Math.round(Math.random()*300);
9     _root["mol_mc"+i]._xscale = scale;
10    _root["mol_mc"+i]._yscale = scale;
11
12 }
13 }

```

- 7) Explanation :

We use a **for** loop to loop th number of molecules that is being generated. The syntax within the for loop is :

```

for (iterant; condition; increment){
//codes
}

```

The variable **i** is initialised to **0** and the condition is smaller than the number that the person typed in within the **molecules_txt** text box. You noticed before the **molecules_txt**, there is a **Number** before that? That is called a **typecast**. You have to do that because anything that is retrieved from a text box is considered to be a string. So you have to convert the result to a numerical value.

Of course we put that in a callback function using the general syntax :

```

callbackInstance.event = function(){
//codes
}

```

In other words, what it means is that if a user clicks and releases **gen_btn**, than something happens. This method is useful if you attach the codes to the Timeline and control any events from there.

Utilising some of the commands in MovieClips

8) Lets take a look at some of the commands of a MovieClip namely - **duplicateMovieClip**

```
molecule_mc.duplicateMovieClip ("mol_mc" + i, i);
```

9) **Explanation :**

You notice that it has 2 parameters? One is the **name** and the other is the **depth** for the new duplicated instance. We want the name to be different for every new instance to be created. So we append a variable(i) to the name of the variable (mol_mc). The depth is just the level in which the new instance movieclip resides on. Bear in mind that each new movieclip has a unique depth level.

10) Another one is **_xscale**

```
_root["mol_mc"+i]._xscale = scale;
```

11) **Explanation :**

This is a property of a movieclip which determines the horizontal scale (percentage) of the movie clip as applied from the registration point of the movie clip. The registration point is usually at the center point.

You notice something interesting about this part is that there is a square bracket? Because of the fact that every instance name is different like **mol_mc1**, **mol_mc2** etc. We have to enclose this in a square bracket. Before the square bracket, we have to insert an identifier, usually the main timeline. The main timeline is usually denoted by **_root**

Removing MovieClips

12) To remove or to reset this simple molecule generator, just type in the following :

```
15 cancel_btn.onRelease = function(){  
16     for (i=0; i<Number(molecules_txt.text); i++) {  
17         _root["mol_mc"+i].removeMovieClip();  
18     }  
19 }
```

13) **Explanation :**

You notice that I place the codes within a callback function on the instance called **cancel_btn**. I loop the same number of times as what I have created. I use another command of MovieClip called **removeMovieClip()**. It removes a movie clip instance created with **duplicateMovieClip()**.