section 2: ActionScript Interfaces

chapter 9: Animation and Interactivity with the Drawing API

In the past, using Flash to create graphics at run-time required at least some form of using predefined symbols and manipulating them with code. Sure, you could take a single pixel image in your Library and do some interesting things with it, but you were rather limited in your ability to create useful, dynamic, graphics if you hadn't drawn them manually beforehand. As an example, one of the first experiments many Flash-users attempt is a drawing program, but all quickly learn that only so much can be accomplished with a 100x100 pixel hairline and duplicated movie clips.

Now – hallelujah – with Flash MX, Macromedia has introduced the ability to create graphics at run-time and nothing will ever be the same again (OK, maybe that's a little melodramatic, but at the very least creating dynamic graphics won't ever be the same, and that's what this section is all about).

Using the new drawing API (Application Programming Interface) and the new movie clip methods createEmptyMovieClip and createTextField, Flash users can now code and create anything from buttons and forms, to drawing applications, to entire interfaces without drawing a single graphic beforehand. Don't believe me? Well, it's true, and we'll spend the next few chapters exploring how.

Coded line drawing

Before we get into more complex applications of the drawing API, let's quickly review some of the methods that you explored in chapter 3:

- **lineStyle** sets the visual properties of the line, such as weight, color, and alpha.
- **moveTo** moves the 'pencil' to new coordinates on the stage without drawing a line between them.
Flash MX Studio

- `lineTo` draws a straight line from the pencil's current position to the given coordinates, while `curveTo` creates a curved line to the given anchor point coordinates by curving towards the control point coordinates.

- Finally, `beginFill` and `endFill` act as bookends for the solid color fills in your shapes, and `clear` wipes the canvas clean.

There is only one more command left to explore, but it's a big one: `beginGradientFill`. We'll look into this fun little method later in the chapter.

OK, enough review. Let's work through a short exercise, using the drawing API to draw lines of random color and size.

### Chaos lines

In this short exercise, we're going to see how just a few short lines of code can create a quick effect (it's `chaosLines.fla` on the CD).

1. Create a new movie, and rename the default layer `actions`. Open the Actions panel and add the following code to frame 1:

   ```actionscript
   pos = {x:0, y:0};
   stageWidth = Stage.width;
   stageHeight = Stage.height;
   ```

   These three lines initialize some variables for our experiment. First, we create a new object called `pos` that contains the properties `x` and `y`, which will be our changing coordinates throughout the movie. Next we look at the properties of the new `Stage` object and place their values into variables. If we didn't do this at the beginning, these values could change during the course of our experiment and our lines would be drawn outside the stage area.

   One thing to note here: when I declare new arrays or objects, I don't use the constructor `new`, but opt for the shorthand. Therefore, a new empty array or an empty object in my code would be, respectively:

   ```actionscript
   myArray = [];
   myObject = {};
   ```

   This will work exactly the same as declaring the objects and arrays like this:

   ```actionscript
   myArray = new Array();
   myObject = new Object();
   ```