Macromedia Flash MX

A Brief Tutorial for
“Programming Usable Interfaces”
Andrew Ko
Versions

- Macromedia **Flash MX**
- **Not** Flash MX 2004 (the newer version)
- Not any older version
- No site license
- PC version: Wean clusters, Cyert Hall
- Mac version: Hunt Library, Cyert Hall
A Beginner’s Book

- *Foundation Macromedia Flash MX*
- Kris Besley
- Comes highly recommended on Amazon
- ~$10 paperback on Amazon and Half.com
- Not a reference, but a very detailed introduction for people unfamiliar with Flash
VB.NET vs. Flash MX

- VB is great for **form-based** applications
- Flash is great for **time-based** applications
- Both VB and Flash are event-based.
What is Flash?

Drawing

Timeline

Stage

Vector Graphics

ActionScript

And lots and lots of terminology...

Intro • The Stage • Vector Graphics • Timelines • Layers • Symbols • ActionScript • Buttons
The Stage

- Contains the objects, images, drawings, buttons, etc.
- Where drawings are created, modified, deleted, etc.
- Things can go outside the stage, but they are clipped when the movie is played.
The Stage

- Modify the stage size, background color, frames per second, ruler units, etc. by going to:

- **Modify → Document...**

![Document Properties dialog box](image-url)
Vector Graphics

All shapes in Flash are made out of points, lines, and curves. Even this one:
Vector Graphics

- Because shapes are made out of points, Flash
- Snaps objects’ edges together
- Snaps objects’ points to each other
- All shapes have an outline and fill color, a line pattern, and a stroke thickness
- These can be changed in the property window

Intro • The Stage • Vector Graphics • Timelines • Layers • Symbols • ActionScript • Buttons
Freeform Lines and Shapes

Pencil

Your freeform stroke is turned into a modifiable shape.

Options that allow you to straighten, smooth, or “ink” (no modification)
Lines, Ovals, and Rectangles

Line

Ovals and rectangles

Options

For rectangles, the “radius” of the corner
Paths of Lines and Curves

Pen

Click once to make a straight line in the path.

Click and drag to make a curve.

Click on a previously made point to close the shape.

Pay close attention to the changes in the cursor with this tool.
Like brushes in other applications, has a **radius** and **shape**.

The options also allow for filling **behind**, **selection**, **inside**...
Erasing Parts of Shapes

Like erasers in other applications, has a radius and shape.

The options also allow for erasing fills, lines, and other things.
The Timeline

• A view of all of the **frames** and **layers** in your **movie**

• Of course, most Flash-based media is interactive, so it's not **exactly** a movie.
Frames

Frame 1 is currently selected

This is a frame

This movie shows 12 frames per second
Frames

- Use the **insert** menu to insert and remove frames.
- We’ll insert 11 frames
The black dot means that frame 1 is a **keyframe**.

The grey frames 2-12 have the same content as the frame 1, the keyframe.

The hollow white rectangle means that it is the end of the span of frames.

A keyframe defines the content of **all frames following it**, up until the **next** keyframe.
Intr

The Stage • Vector Graphics • Timelines • Layers • Symbols • ActionScript • Buttons

Frames

Frame 1, a keyframe, has a black oval on the stage.

All of the grey frames following it have the same black oval as frame 1.

The last frame that with the black oval is frame 12, represented by the white rectangle.

This movie will show this black oval for 12 frames (1 second) and then loop.
Intr • The Stage • Vector Graphics • Timelines • Layers • Symbols • ActionScript • Buttons
If we want to change what the 13th frame displays, we “insert a keyframe”

Inserting actually just changes the selected frame to a keyframe.

This **copies** the previous keyframe’s contents.

We can then change the contents of the new keyframe.
Frames

Now, the first 12 frames have a black oval.

The 13th frame has a red oval of the same size.
Layers

• Just like Illustrator and other applications, Flash supports **layers** of content.

• The layers are part of the timeline.

• Layers can be locked, hidden, and organized in folders.

• *Layers determine the order in which content is displayed for each frame.*
Here I’ve renamed **Layer 1** to **Nose** and added a layer called **Face**.

Then I drew a brown rectangle on the **Face** layer.
Then I added an **Eyes** layer, and drew the eyes and mouth.

Then I drew a the eyes and the mouth.
Symbols

- Symbols are tricky.
- Symbols can also be thought of as a class.
  - A single Button class ➔ many Button instances
- Symbols = Separate Timeline + Stage + Layers
- So they can also be thought of as movies inside of movies.
Types of Symbols

- **Graphic**
  - Operate in sync with the main movie’s timeline.
  - Used for images or reusable animations.

- **Movie Clip**
  - Movie inside a movie. On their own time, not necessarily in sync with the main timeline.
  - Good for interactive things and sounds.
Types of Symbols

- **Buttons**
  - Special kind of Movie Clip symbol that responds to clicks, rollovers and other actions.
  - You define the graphics for each of the various button states, and then assign particular actions to an *instance* of a button.
Making a Symbol

You can make a new, blank symbol by going to Insert → Insert Symbol...

Here, we have selected the eye (by locking the nose and face layers and dragging a box around the eye), so the menu says Convert to Symbol instead.
The Library

Now, the eye appears in the *Library* as a new symbol.

The library contains all of this Flash document’s symbols.
An Instance of a Symbol

Now, the left eye is an instance of the **AnimatedEye** symbol.

After erasing the right eye, we can drag a new instance of the **AnimatedEye** symbol from the library for the right eye.

Now we have two instances of the animated eye symbol, each with their own positions, but sharing the symbol’s timeline, layers, etc.
Naming Instances

It's important to name an instance if you want to refer to it in code.

We name the left eye `leftEye` by selecting the left eye and changing the instance name in the property window.

We do the same for the right eye.
Editing a Symbol

To edit a symbol, double-click on the symbol in the library (either the name or image).

Notice that the timeline, layers, and stage are now specific to the AnimatedEye symbol.

Our view is now inside the main scene. You can return to it by clicking here.
Editing a Symbol

Let’s make the eyes blink every 1/12th of a second.

If we right-click on the 12th frame in the eye’s timeline and select insert frame, it inserts frames in between.

Then we make a keyframe, and change the eye so that it is blinking.
Let’s make the eyes look at the mouse using **ActionScript**, the Flash scripting language.

ActionScript is event-based, like VB.NET, and has two main types of events:

- **Button events** (pressed, released, rollOver...)
- **MovieClip events** (load, enterFrame, mouseMove...)

Each has its own syntax.
ActionScript

• ActionScript code is associated with frames, Movie Clips, and Buttons.

• Just before a frame is displayed, its code is executed.

• The frames following a keyframe share the same code as the keyframe. The keyframe’s code is executed just before it is displayed.
The Action Window

There are two editing modes:

In **expert** mode, you type, and can use the *Intellisense* like pop-up to see the available methods.

In **normal** mode, you drag and drop “Actions” from the area in the left.
The code shown in the Actions window depends on the current selection in the Flash environment.

Individual frames can have ActionScript. This is represented by the ‘a’ in the frame.

All frames that follow a keyframe share the same ActionScript code.

To help you find your code, make a layer in the timeline that is reserved for code only. Then you only have to search in that one layer.
The code shown in the Actions window depends on the current selection in the Flash environment.

Movie clips can also have code, but it can only be movie clip event handlers.

Notice that the movie clip’s code appears when the movie clip is selected.
Following the Mouse

To make the eyes follow the mouse cursor, we will put event handling code “in” each eye.

The Actions window for the left eye instance doesn’t have any code yet.
This event handler will be called when the mouse moves.
Following the Mouse

These two lines define two variables, `xDelta` and `yDelta`.

- `_root._xmouse` refers to the current x position of the mouse, based on the main stage’s coordinate system.
- `xDelta = _root._xmouse - _x`
- `yDelta = _root._ymouse - _y`

`_x` and `_y` refer to the x and y position of the left eye, based on the main stage’s coordinate system.
This defines degrees.

\textbf{atan2()} takes the x and y deltas and returns the degrees in radians.

This math converts the radians to degrees.
Following the Mouse

This sets the rotation of the left eye to the value of degrees.

This -5 corrects for my sloppy drawing.
Intro • The Stage • Vector Graphics • Timelines • Layers • Symbols • ActionScript • Buttons
We’ll edit the symbol by double-clicking on it in the library...

Go to the blink frame of the eye animation...

Open the Action window for the frame...

...and set the rotation of the eye to 0 for that frame.
Create a “blinking” variable that’s **true** when Rudolph is blinking....

Then remember the old rotation so we can restore it after blinking.
In the first frame, we’ll restore the old rotation value.
Then, we’ll avoid rotating the eye if it’s in the middle of blinking.
Finally, we’ll copy the code from the left eye to the right eye.
Intro • The Stage • Vector Graphics • Timelines • Layers • Symbols • ActionScript • Buttons
Buttons

- Buttons are a special type of Symbol, which have frames for each of their states.
- Let’s make Rudolph’s nose a button by converting it to a symbol.
Buttons

A Button has four frames, each defining its appearance and behavior in different states.

**Up** is when the button is not pressed, or released.

**Over** is when the button is hovered over.

**Down** is when the button is pressed.

**Hit** defines the clickable area of the button.
The nose’s **up** frame will be just as it is.
To make the nose’s **over** frame, insert a keyframe.

I’ve filled the nose with brown.

If we don’t make a keyframe, **this changes the up frame to brown as well.**
Down

To make the nose’s **down** frame, insert a keyframe.

I’ve filled the nose with red.
By default, the clickable area is defined by the oval on the screen.
Button Events

To make an event for the button, select the NoseButton instance on the stage, and open the Actions window.

Use on(event) to respond to press, release, rollOver, etc.
Button Events

When pressed, the eyes will goto frame 12 (the blinking frame) and stop playing.

When the nose button is released, the eyes will go to frame 1 and start playing.
Button Events
We’ll remove the keyframe with the red nose, so the nose button is clickable for the whole movie.
Button Events