

MingX Manual

MingX is a wrapper which wraps Ming library in ActiveX object.

Ming is an open-source (LGPL) library which allows you to create SWF ("Flash") format movies. Ming supports almost all of Flash 4's features, including: shapes, gradients, bitmaps (pngs and jpegs), morphs ("shape tweens"), text, buttons, actions, sprites ("movie clips"), streaming mp3, and color transforms.

The reference and examples are taken from <http://ming.sourceforge.net/>. We've made some changes to fit MingX.

Objects

Movie

```
m = new ActiveXObject("Mingx.Movie");  
m.create();
```

Create a new movie object, representing a SWF version 4 movie.

```
m.save(filename);
```

Saves your movie to the named file.

```
i = m.add(c);
```

Add any type of data to a movie. Shapes, text, fonts, etc. must all be added to the movie to make this work. For displayable types (shape, text, button, sprite), this returns a [DisplayItem](#) object, represents the object in a display list. Thus, you can add the same shape to a movie multiple times and get separate DisplayItem objects back for each separate instance.

```
m.remove(i);
```

Remove the object instance from the display list.

```
m.nextFrame();
```

Move to the next frame of the animation.

```
m.setBackground(r, g, b);
```

Set the background color.

```
m.setRate(frameRate);
```

Set the frame rate. Animation will slow down if the player can't render frames fast enough- unless there's a streaming sound, in which case display frames are sacrificed to keep sound from skipping.

```
m.setDimension(width, height);
```

Set the movie's width and height.

```
m.setFrames(numberOfFrames);
```

Set the total number of frames in the animation.

```
m.streamMp3(input);
```

Stream a mp3 file. input is an Input object, you can create it with a mp3 file

```
var input = new ActiveXObject("Mingx.Input");  
input.create("c:\\test.mp3");
```

Shape

```
s = new ActiveXObject("Mingx.Shape");  
s.create();
```

Creates a new shape object.

```
s.setLine(width, r, g, b [, a]);
```

Sets the shape's line style.

```
s.setLine(0, r, g, b [, a]);
```

Removes the shape's line style.

```
f = s.addFill(r, g, b [, a]);
```

Adds a solid fill to the shape's list of fill styles. Returns a [Fill](#) object for use with the [setFill](#) functions described below.

```
f = s.addFill_Bitmap(bitmap [, flags]);
```

Adds a bitmap fill to the shape's fill styles. Returns a [Fill](#) object.

The bitmap argument is a [Bitmap](#) object. The flags argument can be SWFFILL_CLIPPED_BITMAP or SWFFILL_TILED_BITMAP. Default is SWFFILL_TILED_BITMAP.

```
SWFFILL_TILED_BITMAP      = 0x40  
SWFFILL_CLIPPED_BITMAP   = 0x41
```

```
f = s.addFill_Gradient(gradient [, flags]);
```

Adds a gradient fill to the shape's fill styles. Returns a [Fill](#) object.

The gradient argument is a [Gradient](#) object. The flags argument can be SWFFILL_RADIAL_GRADIENT or SWFFILL_LINEAR_GRADIENT. Default is SWFFILL_LINEAR_GRADIENT.

```
SWFFILL_LINEAR_GRADIENT = 0x10  
SWFFILL_RADIAL_GRADIENT = 0x12
```

```
s.setLeftFill(fill);  
s.setRightFill(fill);
```

What this nonsense is about is, every edge segment borders at most two fills. When rasterizing the object, it's pretty handy to know what those fills are ahead of time, so the swf format requires these to be specified. `setLeftFill` sets the fill on the left side of the edge- that is, on the interior if you're defining the outline of the shape in a counter-clockwise fashion. The fill object is a [Fill](#) object returned from one of the [addFill](#) functions above.

This seems to be reversed when you're defining a shape in a morph, though. If your browser crashes, just try setting the fill on the other side.

```
s.movePenTo(x, y);
```

Move the shape's pen to (x, y) in the shape's coordinate space.

```
s.movePen(dx, dy);
```

Add dx to pen's current x position, dy to y.

```
s.drawLineTo(x, y)
```

Draw a line (using the current line style) from the current pen position to point (x, y) in the shape's coordinate space.

```
s.drawLine(dx, dy);
```

Draw a line (using the current line style) from the current pen position to displacement (dx, dy).

```
s.drawCurveTo(controlx, controly, anchorx, anchory);
```

Draw a quadratic curve (using the current line style) from the current pen position to (anchorx, anchory) using (controlx, controly) as a control point. That is, head towards the control point, then smoothly turn to the anchor point. Wait, maybe a diagram will help:



```
s.drawCurve(controldx, controldy, anchordx, anchordy);
```

Draw a quadratic curve (using the current line style) from the current pen position to displacement (anchordx, anchordy) using displacement (controldx, controldy) as a control point.

DisplayItem

Here's where all the animation takes place. After you define a shape, a text object, a sprite, or a button, you [add](#) it to the movie, and then use the returned object to move, rotate, scale, or skew the thing.

```
i.moveTo(x, y);
```

Move object to (x, y) in global coordinates.

```
i.move(x, y);
```

Displace object by (x, y) in global coordinates.

```
i.scaleTo(x, y);
```

Set scale to x in the x-direction, y in the y-direction.

```
i.scale(x, y);
```

Multiply scale by x in the x-direction, y in the y-direction.

```
i.rotateTo(degrees);
```

Set rotation to degrees degrees.

```
i.rotate(degrees)
```

Rotate by degrees degrees.

```
i.skewXTo(x);
```

Set x-skew x. 1.0 is a 45-degree forward slant. More is more forward, less is more backward.

```
i.skewX(x)
```

Add x to current x-skew.

```
i.skewYTo(x);
```

Set/add y-skew y. 1.0 is a 45-degree upward slant. More is more upward, less is more downward.

```
i.skewY(x)
```

Set/add y-skew y. 1.0 is a 45-degree upward slant. More is more upward, less is more downward.

```
i.setDepth(depth);
```

Set z-order to depth. Depth defaults to the order in which instances are created (by adding a shape/text to a movie)- newer ones are on top of older ones. If two objects are given the same depth, only the later-defined one can be moved.

```
i.remove();
```

Remove this object from the movie's display list.

```
i.setName(name);
```

Set the object's name to name, for targeting with action script. Only useful on sprites.

```
i.setRatio(ratio);
```

Set the object's ratio to ratio. Obviously only useful for morphs.

```
i.addColor(r, g, b [, a])
```

Adds the given color to this item's color transform.

```
i.multColor(r, g, b [, a]);
```

Multiplies the item's color transform by the given values.

Gradient

```
g = new ActiveXObject("Mingx.Gradient");  
g.create();
```

Create a new Gradient object.

After you've added the entries to your gradient, you can use the gradient in a shape fill with the Shape [addFill](#) method.

```
g.addEntry(ratio, r, g, b [, a]);
```

Add an entry to the gradient list. ratio is a number between 0 and 1 indicating where in the gradient this color appears. Thou shalt add entries in order of increasing ratio.

Bitmap

```
b = new ActiveXObject("Mingx.Bitmap");  
b.create(input[, alpha]);
```

Create a new Bitmap object from the given Jpeg file. `alphafilename` indicates a MSK file to be used as an alpha mask for a Jpeg image.

Note: we can only deal with baseline (frame 0) jpegs, no baseline optimized or progressive scan jpegs!

`input` and `alpha` are Input objects. You can create an Input object from a jpg file like this

```
input = new ActiveXObject("Mingx.Input");  
input.create("c:\\test.jpg");
```

```
width = b.width;
```

Returns the bitmap's width in pixels.

```
height = b.height;
```

Returns the bitmap's height in pixels.

Fill

The Fill object allows you to transform (scale, skew, rotate) bitmap and gradient fills. Fill objects are created by the Shape [addFill](#) methods.

```
f.moveTo(x, y);
```

Move fill origin to (x,y) in global coordinates.

```
f.scaleTo(x, y);
```

Set fill scale to x in the x-direction, y in the y-direction.

```
f.rotateTo(degrees);
```

Set fill rotation to degrees degrees.

```
f.skewXTo(x);
```

Set fill x-skew to x. 1.0 is a 45-degree forward slant. More is more forward, less is more backward.

```
f.skewYTo(x);
```

Set fill y-skew to y. 1.0 is a 45-degree upward slant. More is more upward, less is more downward.

Morph

Also called a "shape tween". This thing lets you make those tacky twisting things that make your computer choke. Oh, joy!

```
m = new ActiveXObject("Mingx.Morph");  
m.create();
```

Create a new Morph object.

```
s = m.shape1;
```

Get an object to the morph's starting shape. Returns a Shape object.

```
s = m.shape2;
```

Get an object to the morph's ending shape. Returns a Shape object.

Text

```
t = new ActiveXObject("Mingx.Text");  
t.create();
```

Create a new text object, fresh for manipulatⁿ.

```
t.font = font;
```

Set the current font to font. The font must be created from a fdb file.

```
t.height = height;
```

Set the current font height to height. Default is 240.

```
t.spacing = spacing;
```

Set the current font spacing to spacing. Default is 1.0. 0 is all of the letters written at the same point. This doesn't really work that well because it inflates the advance across the letter, doesn't add the same amount of spacing between the letters. I should try and explain that better, proly. Or just fix the damn thing to do constant spacing. This was really just a way to figure out how letter advances work, anyway.. So nyah.

```
t.setColor(r, g, b [, a]);
```

Change the current text color.

```
t.moveTo(x, y);
```

Move the pen (or cursor, if that makes more sense) to (x,y) in text object's coordinate space. If either is zero, though, value in that dimension stays the same. Annoying, should be fixed.

```
t.text = string;
```

Draw the string string at the current pen (cursor) location. Pen is at the baseline of the text; i.e., ascending text is in the -y direction.

```
width = t.getWidth(string);
```

Returns the rendered width of the given string at the text object's current font, scale, and spacing settings.

Font

```
f = new ActiveXObject("Mingx.Font");  
f.create(name);
```

If name is the name of an FDB file (i.e., it ends in ".fdb"), load the font definition found in said file. Otherwise, create a browser-defined font reference.

FDB ("font definition block") is a very simple wrapper for the SWF DefineFont2 block which contains a full description of a font. You can create FDB file with **makefdb.exe**.

Browser-defined fonts don't contain any information about the font other than its name. It is assumed that the font definition will be provided by the movie player. The fonts `_serif`, `_sans`, and `_typewriter` should always be available.

```
width = f.getWidth(string);
```

Returns the width of the given string at the font's default scaling. You'll probably want to use the `Text` version of this method which uses the text object's scale.

TextField

Text Fields are less flexible than Text objects- they can't be rotated, scaled non-proportionally, or skewed, but they can be used as form entries, and they can use browser-defined fonts.

```
t = new ActiveXObject("Mingx.TextField");  
t.create([flags]);
```

Create a new text field object.

The optional flags change the text field's behavior:

`SWFTEXTFIELD_NOEDIT = 1<<3` indicates that the field shouldn't be user-editable

`SWFTEXTFIELD_PASSWORD = 1<<4` obscures the data entry

`SWFTEXTFIELD_DRAWBOX = 1<<11` draws the outline of the textfield

`SWFTEXTFIELD_MULTILINE = 1<<5` allows multiple lines

`SWFTEXTFIELD_WORDWRAP = 1<<6` allows text to wrap

`SWFTEXTFIELD_NOSELECT = 1<<12` makes the field non-selectable

Flags are combined with the bitwise `|` operation.

```
t.font = font;
```

Set the text field font to the font `font`.

```
t.setBounds(width, height);
```

Set the text field width and height. If you don't set the bounds yourself, Ming makes a poor guess at what the bounds are.

```
t.align(alignment);
```

Set the text field alignment. Legit values are:

```
SWFTEXTFIELD_ALIGN_LEFT = 0
```

```
SWFTEXTFIELD_ALIGN_RIGHT = 1
```

```
SWFTEXTFIELD_ALIGN_CENTER = 2
```

```
SWFTEXTFIELD_ALIGN_JUSTIFY = 3
```

```
t.height = height;
```

Set the font height of this text field font to the given height. Default is 240.

```
t.leftMargin = width;
```

Set the left margin width of the text field. Default is 0.

```
t.rightMargin = width;
```

Set the right margin width of the text field. Default is 0.

```
t.setMargins(left, right);
```

Set both margins at once, for the man on the go.

```
t.indentation = width;
```

Set the indentation of the first line in the text field.

```
t.lineSpacing = height;
```

Set the line spacing of the text field. Default is 40.

```
t.setColor(r, g, b [, a]);
```

Set the color of the text field. Default is fully opaque black.

```
t.name = name;
```

Set the variable name of this text field, for form posting and action scripting purposes.

```
t.text = string;
```

Concatenate the given string to the text field.

Sprite

Also known as a "movie clip", this allows one to create objects which are animated in their own timelines. Hence, the sprite has most of the same methods as the movie.

```
s = new ActiveXObject("Mingx.Sprite");
```

```
s.create();
```

Create a new Sprite object.

```
i = s.add(c);
```

Add a shape, text, a button, an action, or another sprite to the sprite.

For displayable types (shape, text, button, sprite), this returns a DisplayItem object to the object in a display list.

```
s.remove(i);
```

Remove the object instance from the display list.

```
s.nextFrame();
```

Move to the next frame of the animation.

```
s.setFrames(numberOfFrames);
```

Set the total number of frames in the sprite's animation.

Button

```
b = new ActiveXObject("Mingx.Button");  
b.create();
```

Create a new Button object.

```
b.addShape(shape, flags);
```

Add the shape shape to this button. The following flags are valid:

```
SWFBUTTON_UP = 1<<0  
SWFBUTTON_OVER = 1<<1  
SWFBUTTON_DOWN = 1<<2  
SWFBUTTON_HIT = 1<<3
```

SWFBUTTON_HIT isn't ever displayed, it defines the hit region for the button. That is, everywhere the hit shape would be drawn is considered a "touchable" part of the button.

```
b.up = shape;
```

Alias for addShape(shape, SWFBUTTON_UP)

```
b.over = shape;
```

Alias for addShape(shape, SWFBUTTON_OVER)

```
b.down = shape;
```

Alias for addShape(shape, SWFBUTTON_DOWN)

```
b.hit = shape;
```

Alias for addShape(shape, SWFBUTTON_HIT)

```
b.addAction(action, flags);
```

Add the action action to this button for the given conditions. The following flags are valid:

```
SWFBUTTON_MOUSEOVER = 1<<0  
SWFBUTTON_MOUSEOUT = 1<<1  
SWFBUTTON_MOUSEUP = 1<<3  
SWFBUTTON_MOUSEUPOUTSIDE = 1<<6  
SWFBUTTON_MOUSEDOWN = 1<<2  
SWFBUTTON_DRAGOUT = (1<<4) | (1<<8)  
SWFBUTTON_DRAGOVER = (1<<5) | (1<<7)
```

```
b->action = action;
```

Sets the action to be performed when the button is clicked. Alias for addAction(shape, SWFBUTTON_MOUSEUP)

Action

```
a = new ActiveXObject("Mingx.Action");  
a.create(script);
```

Compiles the given script into an Action object. Ming now compiles Flash 5 action script.

Examples

1.Drawing shapes

```
s = new ActiveXObject("Mingx.Shape");
s.create();
s.setLine(4, 0x7f, 0, 0);
s.setRightFill(s.addFill(0xff, 0, 0));
s.movePenTo(10, 10);
s.drawLineTo(310, 10);
s.drawLineTo(310, 230);
s.drawCurveTo(10, 230, 10, 10);

m = new ActiveXObject("Mingx.Movie");
m.create();
m.setDimension(320, 240);
m.setRate(12.0);
m.add(s);
m.nextFrame();

m.save("c:\\1.swf");
```

2.Making text

```
f = new ActiveXObject("Mingx.Font");
f.create("e:\\1.fdb");
t = new ActiveXObject("Mingx.Text");
t.create();
t.font = f;
t.setColor(0xff, 0xff, 0);
t.height = 60;
t.text = "fnar! fnar!";

m = new ActiveXObject("Mingx.Movie");
m.create();
m.setDimension(320, 240);

i = m.add(t);
i.moveTo(160 - t.getWidth("fnar! fnar!") / 2, 120 + t.ascent()/2);
```

```
m.save("c:\\1.swf");
```

3. Drawing Glyphs

```
s = new ActiveXObject("Mingx.Shape");
s.create();
f1 = s.addFill(0xff, 0, 0);
f2 = s.addFill(0xff, 0x7f, 0);
f3 = s.addFill(0xff, 0xff, 0);
f4 = s.addFill(0, 0xff, 0);
f5 = s.addFill(0, 0, 0xff);

f = new ActiveXObject("Mingx.FOnt");
f.create('e:\\1.fdb');

s.setRightFill(f1);
s.setLine(2, 0x7f, 0, 0);
s.drawGlyph(f, '!');
s.movePen(f.getWidth('!'), 0);

s.setRightFill(f2);
s.setLine(2, 0x7f, 0x3f, 0);
s.drawGlyph(f, '#');
s.movePen(f.getWidth('#'), 0);

s.setRightFill(f3);
s.setLine(2, 0x7f, 0x7f, 0);
s.drawGlyph(f, '%');
s.movePen(f.getWidth('%'), 0);

s.setRightFill(f4);
s.setLine(2, 0, 0x7f, 0);
s.drawGlyph(f, '*');
s.movePen(f.getWidth('*'), 0);

s.setRightFill(f5);
s.setLine(2, 0, 0, 0x7f);
s.drawGlyph(f, '@');

m = new ActiveXObject("Mingx.Movie");
m.create();
m.setDimension(320, 240);
i = m.add(s);
i.scaleTo(2.0, 2.0);
```

```
i.moveTo(160-f.getWidth("!#%*@"), 120+f.ascent());
```

```
m.save("c:\\1.swf");
```

4. Animation

```
argv = "ming!";
```

```
f = new ActiveXObject("Mingx.Font");
```

```
f.create("e:\\1.fdb");
```

```
m = new ActiveXObject("Mingx.Movie");
```

```
m.create();
```

```
m.setRate(24.0);
```

```
m.setDimension(320, 240);
```

```
m.setBackground(0xff, 0xff, 0xff);
```

```
// functions with huge numbers of arbitrary
```

```
// arguments are always a good idea! Really!
```

```
function text(r, g, b, a, rot, x, y, scale, string)
```

```
{
```

```
    t = new ActiveXObject("Mingx.Text");
```

```
    t.create();
```

```
    t.font = f;
```

```
    t.setColor(r, g, b, a);
```

```
    t.height = 96;
```

```
    t.moveTo(-(t.getWidth(string))/2, 32); //f.getAscent()/2);
```

```
    t.text = string;
```

```
    i = m.add(t);
```

```
    i.x = x;
```

```
    i.y = y;
```

```
    i.rot = rot;
```

```
    i.s = scale;
```

```
    i.rotateTo(rot);
```

```
    i.scale(scale, scale);
```

```
// but the changes are local to the function, so we have to
```

```
// return the changed object. kinda weird..
```

```
    return i;
```

```
}
```

```

function step(i)
{
    oldrot = i.rot;
    i.rot = 19*i.rot/20;
    i.x = (19*i.x + 160)/20;
    i.y = (19*i.y + 120)/20;
    i.s = (19*i.s + 1.0)/20;

    i.rotateTo(i.rot);
    i.scaleTo(i.s, i.s);
    i.moveTo(i.x, i.y);

    return i;
}

// see? it sure paid off in legibility:

i1 = text(0xff, 0x33, 0x33, 0xff, 900, 160, 120, 3, argv);
i2 = text(0x00, 0x33, 0xff, 0x7f, -560, 160, 120, 4, argv);
i3 = text(0xff, 0xff, 0xff, 0x9f, 180, 160, 120, 1, argv);

// (.that was sarcasm.)

for(i=1; i<=100; ++i)
{
    i1 = step(i1);
    i2 = step(i2);
    i3 = step(i3);

    m.nextFrame();
}

m.save("c:\\1.swf");

```

5.Sprite

```

s = new ActiveXObject("Mingx.Shape");
s.create();
s.setRightFill(s.addFill(0xff, 0, 0));
s.movePenTo(-50,-50);
s.drawLineTo(50,-50);
s.drawLineTo(50,50);
s.drawLineTo(-50,50);
s.drawLineTo(-50,-50);

```

```

p = new ActiveXObject("Mingx.Sprite");
p.create();
i = p.add(s);

for(j=0; j<17; ++j)
{
    p.nextFrame();
    i.rotate(5);
}
p.nextFrame();

m = new ActiveXObject("Mingx.Movie");
m.create();
i = m.add(p);
i.moveTo(160,120);
i.setName("blah");

m.setBackground(0xff, 0xff, 0xff);
m.setDimension(320,240);

m.save("c:\\1.swf");

```

6.Morph

```

p = new ActiveXObject("Mingx.Morph");
p.create();

s = p.shape1();
s.setLine(0,0,0,0);

/* Note that this is backwards from normal shapes (left instead of right).
   I have no idea why, but this seems to work.. */

s.setLeftFill(s.addFill(0xff, 0, 0));
s.movePenTo(-1000,-1000);
s.drawLine(2000,0);
s.drawLine(0,2000);
s.drawLine(-2000,0);
s.drawLine(0,-2000);

s = p.shape2();
s.setLine(60,0,0,0);
s.setLeftFill(s.addFill(0, 0, 0xff));

```

```

s.movePenTo(0,-1000);
s.drawLine(1000,1000);
s.drawLine(-1000,1000);
s.drawLine(-1000,-1000);
s.drawLine(1000,-1000);

m = new ActiveXObject("Mingx.Movie");//SWFMovie();
m.create();
m.setDimension(3000,2000);
m.setBackground(0xff, 0xff, 0xff);

i = m.add(p);
i.moveTo(1500,1000);

for(r=0.0; r<=1.0; r+=0.1)
{
    i.setRatio(r);
    m.nextFrame();
}

m.save("c:\\1.swf");

```

7.Gradient

```

m = new ActiveXObject("Mingx.Movie");
m.create();
m.setDimension(320, 240);

s = new ActiveXObject("Mingx.Shape");
s.create();

// first gradient- black to white
g = new ActiveXObject("Mingx.Gradient");
g.create();
g.addEntry(0, 0.0, 0, 0, 0xff);
g.addEntry(1.0, 0xff, 0xff, 0xff);

f = s.addFill_Gradient(g, 0x10);
f.scaleTo(0.17,0.17);
f.moveTo(160, 120);
s.setRightFill(f);
s.drawLine(320, 0);
s.drawLine(0, 240);
s.drawLine(-320, 0);
s.drawLine(0, -240);

```

```

m.add(s);

s = new ActiveXObject("Mingx.Shape");
s.create();

// second gradient- radial gradient from white to red to transparent
g = new ActiveXObject("Mingx.Gradient");
g.create();
g.addEntry(0.0, 0xff, 0, 0, 0xff);
g.addEntry(1.0, 0xff, 0, 0, 0);

f = s.addFill_Gradient(g, 0x12);
f.scaleTo(0.12, 0.12);
f.moveTo(160, 120);
s.setRightFill(f);
s.drawLine(320, 0);
s.drawLine(0, 240);
s.drawLine(-320, 0);
s.drawLine(0, -240);

m.add(s);

m.save("c:\\1.swf");

```

8. Mouse tracking

```

m = new ActiveXObject("Mingx.Movie");
m.create();
m.setDimension(320, 240);
m.setBackground(0x00, 0x00, 0x00);

action = new ActiveXObject("Mingx.Action");
action.create("_quality = 'LOW';frames._visible = false;sparks = [];");
m.add(action);

g = new ActiveXObject("Mingx.Gradient");
g.create();
g.addEntry(0, 0xff, 0xff, 0xff, 0xff);
g.addEntry(0.1, 0xff, 0xff, 0xff, 0xff);
g.addEntry(0.5, 0xff, 0xff, 0xff, 0x5f);
g.addEntry(1.0, 0xff, 0xff, 0xff, 0);

```

```

// gradient shape thing
s = new ActiveXObject("Mingx.Shape");
s.create();
f = s.addFill_Gradient(g, 0x12);
f.scaleTo(0.03, 0.03);
s.setRightFill(f);
s.movePenTo(-30, -30);
s.drawLine(60, 0);
s.drawLine(0, 60);
s.drawLine(-60, 0);
s.drawLine(0, -60);

// need to make this a sprite so we can multColor it
p = new ActiveXObject("Mingx.Sprite");
p.create();
p.add(s);
p.nextFrame();

// put the shape in here, each frame a different color
q = new ActiveXObject("Mingx.Sprite");
q.create();
action = new ActiveXObject("Mingx.Action");
action.create("gotoFrame(random(7)+1); stop()");
q.add(action);
i = q.add(p);

i.multColor(1.0, 1.0, 1.0);
q.nextFrame();
i.multColor(1.0, 0.5, 0.5);
q.nextFrame();
i.multColor(1.0, 0.75, 0.5);
q.nextFrame();
i.multColor(1.0, 1.0, 0.5);
q.nextFrame();
i.multColor(0.5, 1.0, 0.5);
q.nextFrame();
i.multColor(0.5, 0.5, 1.0);
q.nextFrame();
i.multColor(1.0, 0.5, 1.0);
q.nextFrame();

// finally, this one contains the action code
p = new ActiveXObject("Mingx.Sprite");

```

```

p.create();
i = p.add(q);
i.setName('frames');
action = new ActiveXObject("Mingx.Action");
action.create("dx = _root.dx/3 + random(10)-5;dy = _root.dy/3;x = _root._xmouse;y = _root._ymouse;alpha =
100;");
p.add(action);
p.nextFrame();
action = new ActiveXObject("Mingx.Action");
action.create("this._x = x;this._y = y;this._alpha = alpha;x += dx;y += dy;dy += 3;alpha -= 8;");
p.add(action);
p.nextFrame();

action = new ActiveXObject("Mingx.Action");
action.create("prevFrame(); play();");
p.add(action);
p.nextFrame();

i = m.add(p);
i.setName('frames');
m.nextFrame();

action = new ActiveXObject("Mingx.Action");
action.create("dx = _xmouse - lastx;dy = _ymouse - lasty;lastx = _xmouse;lasty = _ymouse;if(++num == 11)
num = 1;if(sparks[num]) removeMovieClip(sparks[num]);duplicateMovieClip('frames', 'char'+num,
num);sparks[num] = this['char'+num];");
m.add(action);

m.nextFrame();
action = new ActiveXObject("Mingx.Action");
action.create("prevFrame(); play();");
m.add(action);

m.save("c:\\1.swf");

```