Installation

After command:

#include "lmc_tween.as" for ActionScript 2.0 (runs fine on Flash Player 6) or
#include "lmc_tween_as1.as" for ActionScript 1.0 you can use following methods for every MovieClip:

MovieClip.tween()
MovieClip.stopTween()
MovieClip.isTweening()
MovieClip.getTweens()
MovieClip.lockTween()
MovieClip.unlockTween()
MovieClip.isTweenLocked()
MovieClip.alphaTo()
MovieClip.brightnessTo()
MovieClip.brightOffsetTo()
MovieClip.colorTo()
MovieClip.colorTransformTo()
MovieClip.contrastTo()
MovieClip.frameTo()
MovieClip.scaleTo()
MovieClip.slideTo()
MovieClip.rotateTo()

Additional Notes:

If you plan to use this prototypes in your own class, to extend MovieClip class and prevent from compile time errors, you must manully replace the file "MovieClip.as" in your Flash MX 2004\[language]\First Run\Classes directory with file in this location:
WinXP: C:\Documents and Settings\[username]\Local Settings\Application Data\Macromedia\Flash MX 2004\[language]\Configuration\Shared\Zigo
Win9X: C:\Windows\Application Data\Macromedia\Flash MX 2004\[language]\Configuration\Shared\Zigo
Mac: HD:Users:[username]:Library:Application Support:Macromedia Flash MX 2004:[language]:Configuration:Shared:Zigo

Additional Info:

Easing Equations

MovieClip.tween()

Availability

Flash Player 6.

Usage

```
my_mc.tween(property, pEnd, seconds, animType, delay, callback, extra1, extra2)
```

Parameters

property property(ies) to be tweened (string, array of strings)

pEnd end property value(s) (number, array of numbers)

seconds seconds to reach the end value, /duration of tween/ (number) defaults to 2

animType animation equation type (string, function or object from custom easing tool) defaults to "easeOutExpo"

string as animType

You can use following strings as types of animation (easing equations by Robert Penner)

"linear",
"easeInQuad","easeOutQuad","easeInOutQuad","easeOutInQuad"
"easeInCubic","easeOutCubic","easeInOutCubic","easeOutInCubic"
"easeInQuart","easeOutQuart","easeInOutQuart","easeOutInQuart"
"easeInQuint","easeOutQuint","easeInOutQuint","easeOutInQuint"
"easeInSine","easeOutSine","easeInOutSine","easeOutInSine"
"easeInExpo","easeOutExpo","easeInOutExpo","easeOutInExpo"
"easeInCirc","easeOutCirc","easeInOutCirc","easeOutInCirc"
"easeInElastic","easeOutElastic","easeInOutElastic","easeOutInElastic"
"easeInBack","easeOutBack","easeInOutBack","easeOutInBack"
"easeInBounce","easeOutBounce","easeInOutBounce""easeOutInBounce"

example:
my_mc.tween("_x",100,3,"easeOutElastic")

function as animType:

you can use easing function generator from Timothee Groleau to generate your function: e.g.:

```
waveEasing = function(t,b,c,d){
    // ... code from generator
};
my_mc.tween("_x",100,3,waveEasing);
```

object as animType:

you can use custom easing tool from menu window->other panels ->custom easing tool
object must have properties pts (list of control point),ease (function) e.g.:

```
customEasing = {};
// ... copy and paste here generated code
my_mc.tween("_x",100,3,customEasing);
```

*this easing is bit slower than previous two methods

delay delay in seconds to start animation (number) defaults to 0

this parameter allows you create cool sequential tweens:

my_mc.('_x', 200,0.5);
my_mc.('_x', 400,0.5,'easeoutcirc',0.5)
my_mc.tween('_width', 300,1,'easeoutelastic',1)*;*
my_mc.tween('_height', 300,1,'easeoutelastic',2);
my_mc.colorTo(0xFF0000,1,'easeinexpo',3);

callback function to be called when finished (function or object with scope, func and args or string)

function as callback

```
function onEnd(){
trace("onEnd");
}
my_mc.tween("_x",100,1,"linear",0,onEnd);

// scope of function is my_mc._parent
```

object as callback

You can pass as callback object with properties

func - function to be called when tween is finished
scope - scope of function (this in called function)
args - array of arguments passed to function

updfunc - reference to function to be called on every update
updscope - scope of update function (this object)
updargs - array of arguments passed to update function

startfunc - reference to function to be called on start of tween
startscope - scope of start function (this object)
startargs - array of arguments passed to start function

* internal mechanism is: func.apply(scope,args)

```
// on _root


game={};
game.players = ["john","steve"];
game.showScore = function(id, score){
trace("(this==_root.game) is "+(this==_root.game));
trace(this.players[id] + " has " + score + " points");
}

// somewhere in nested movieclip
```

```
var callback = {scope: _root.game, func: _root.game.showScore, args:[1,39]};
my_mc.tween("_x",100,1,"linear",0,callback);
/* or in 1 line:
my_mc.tween("_x",100,1,"linear",0,{scope: _root.game, func:
_root.game.showScore, args:[1,39]});
*/

//output after finishing tween:

(this==_root.game) is true
steve has 39 points
```

string as callback

callbacks can be too defined as strings

my_mc.tween("_x",100,1,"linear",0,'_root.gotoAndPlay(8)');

it is very problematic determine type of primitive parameters(number, string, boolean), in this case is 8 string

for save type of passed argument use references:

function callMe(my_obj, my_nr, my_bool) {
trace(my_obj +">> typeof(my_obj) is "+ typeof(my_obj));
trace(my_nr +">> typeof(my_nr) is "+ typeof(my_nr));
trace(my_bool +">> typeof(my_bool) is "+ typeof(my_bool));
}

test_obj = {name: "test", id: 10};
test_bool = true;
test_nr = 99;

my_mc.tween("_x",100,1,"linear",0,'_root.callMe(test_obj,test_nr,test_bool)');

Do not add spaces between argumets

extra1 optional animation parameter.
means AMPLITUDE (a) when animType *elastic
means OVERSHOOT AMMOUNT (s) when animType *back

extra2 optional animation parameter.
means PERIOD (p) when animType = *elastic

Returns

Nothing

MovieClip.stopTween()

Availability

Flash Player 6.

Usage

```
my_mc.stopTween(props)
```

Parameters

props A string or array of strings indicating properties to be stopped
if you will stop tweening of colorTransform (methods colotTo, brightnessTo, colorTransformTo) use "_ct_" as props parameter

Returns

Nothing.

Description

method, stops tweening of *props* , without passing *props* function stops all tweens in movieclip

example:

```
my_mc.tween(["_x","_width"],[200,0],10);

my_mc.onMouseDown = function(){
 this.stopTween("_x");     // stops _x movement
 //this.stopTween();     // stops all movemets
}
```

MovieClip.isTweening()

Availability

Flash Player 6.

Usage

```
my_mc.isTweening()
```

Parameters

none

Returns

Boolean value indicating that movieclip is tweening

Description

use this method if you will check if your movieclip is being tweened
example:

```
my_mc.tween(["_y","_alpha"],[0,100],5);
_root.onMouseDown = function(){
 if(my_mc.isTweening()){
    my_mc.stopTween();
    trace("my_mc was tweened and now is stoped");
 }
}
```

MovieClip.getTweens()

Availability

Flash Player 6.

Usage

```
my_mc.getTweens()
```

Parameters

none

Returns

Number (count) of running tweening properties

Description

example:

```
my_mc.tween(["_x","_y"],[0,0]);
trace(my_mc.getTweens()); // output 2
```

MovieClip.lockTween()

Availability

Flash Player 6.

Usage

```
my_mc.lockTween()
```

Parameters

none

Returns

nothing

Description

method, disallows performing tweens with movieclip (useful with buttons)
to unlock use .unlockTween() method

following calling of tween method do not perform nothing, because my_mc is locked :

my_mc.lockTween();
my_mc.tween(["_x","_y"],[0,0]);

MovieClip.unlockTween()

Availability

Flash Player 6.

Usage

```
my_mc.unlockTween()
```

Parameters

none

Returns

Nothing

Description

method, allows performing tweens with movieclip (for movieclips that was locked with lock)
Opossite of .lockTween() method

in following example is performed only the second tween command:

```
my_mc.lockTween();
my_mc.tween(["_x","_y"],[0,0]);
my_mc.unlockTween();
my_mc.tween("_width",300);
```

MovieClip.isTweenLocked()

Availability

Flash Player 6.

Usage

```
my_mc.isTweenLocked()
```

Parameters

none

Returns

Bolean value indicating if movieclip can perform tweens

Description

example:

```
trace(my_mc.isTweenLocked()); // false
my_mc.lockTween();
trace(my_mc.isTweenLocked()); // true
my_mc.unlockTween();
trace(my_mc.isTweenLocked()); // false
```

MovieClip.alphaTo()

Availability

Flash Player 6.

Usage

```
my_mc.alphaTo(alpha, seconds, animtype, delay, callback, extra1, extra2)
```

Parameters

alpha end value ofmovieclip _alpha property
all other parameters work as in tween() method

Returns

None

Description

shortcut method; tweens _alpha. Following commands are identical:

```
my_mc.alphaTo(20,1);
my_mc.tween("_alpha",20,1);
```

MovieClip.brightnessTo()

Availability

Flash Player 6.

Usage

```
my_mc.brightnessTo(brightness, seconds, animtype, delay, callback, extra1, extra2)
```

Parameters

brightness A number between -100 (black) and 100 (white) that represents brightness of movieclip

all other parameters do the same as in tween() method

Returns

Nothing.

Description

Following example changes brightness of my_mc to 80 in 3 seconds with quadratic easing

```
my_mc.brightnessTo(80,3,"easeOutQuad");
```

MovieClip.brightOffsetTo()

Availability

Flash Player 6.

Usage

```
my_mc.brightOffsetTo(percent, seconds, animtype, delay, callback, extra1, extra2)
```

Parameters

`percent` A number that represents the brightOffset of movieclip
0 : normal
100 : to white
-100 : to black

all other parameters do the same as in tween() method

Returns

Nothing.

Description

Following example changes tint of my_mc to brightOffset 100 in 2 seconds

```
my_mc.brightOffsetTo(100,2,"easeOutSine");
```

MovieClip.colorTo()

Availability

Flash Player 6.

Usage

```
my_mc.colorTo(color, seconds, animtype, delay, callback, extra1, extra2)
```

Parameters

color A number that represents the RGB numeric value for the color (tint)

all other parameters do the same as in tween() method

Returns

Nothing.

Description

Following example changes tint of my_mc to red (0xFF0000) in 2.5 seconds with easeOutSine effect

```
my_mc.colorTo(0xFF0000,2.5,"easeOutSine");
```

MovieClip.colorTransformTo()

Availability

Flash Player 6.

Usage

```
my_mc.colorTransformTo(ra, rb, ga, gb, ba, bb, aa, ab, seconds, animtype, delay, callback, extra1, extra2)
```

Parameters

ra, rb, ga, gb, ba, bb, aa, ab - use like Color.setTransform

all other parameters do the same as in tween() method

Returns

Nothing.

MovieClip.contrastTo()

Availability

Flash Player 6.

Usage

```
my_mc.contrastTo(percent, seconds, animtype, delay, callback, extra1, extra2)
```

Parameters

percent A number that represents the contrast value for the movieclip
0 : no contrast
100 : normal
100 and more : high contrast
-100 : invert colors

all other parameters do the same as in tween() method

Returns

Nothing.

Description

Change contrast of my_mc to 200 in 2 seconds with easeOutSine effect

```
my_mc.contrastTo(200,2,"easeOutSine");
```

```
Change contrast of my_mc to -100 (invert colors) in 1 second
```

```
my_mc.contrastTo(-100,1,"easeOutSine");
```

MovieClip.frameTo()

Availability

Flash Player 6.

Usage

```
my_mc.frameTo(frame, seconds, animtype, delay, callback, extra1, extra2)
```

Parameters

frame end frame of movieclip timeline
all other parameters work as in tween() method

Returns

None

Description

shortcut method; tweens frames in timeline, my_mc in folowing example must have 20 frames

```
// tween timeline to frame 20 in 1 second with easeoutbounce
my_mc.frameTo(20,1,"easeoutbounce");
// tween timeline back to frame 1 in 0.5 seconds
my_mc.frameTo(1,.5,"linear",1)
```

MovieClip.scaleTo()

Availability

Flash Player 6.

Usage

```
my_mc.scaleTo(size, seconds, animtype, delay, callback, extra1, extra2)
```

Parameters

size A number as _xscale and _yscale

all other parameters do the same as in tween() method

Returns

Nothing.

Description

Shortcut method, following commands are identical (scaling my_mc to 50 % in 0.5 seconds):

```
my_mc.scaleto(50,0.5);
my_mc.tween(["_xscale","_yscale"],[50,50],0.5);
```

MovieClip.slideTo()

Availability

Flash Player 6.

Usage

```
my_mc.slideTo(x,y, seconds, animtype, delay, callback, extra1, extra2);
```

Parameters

x, y numbers specifing end _x and _y properties of movieclip

all other parameters do the same as in tween() method

Returns

Nothing.

Description

Shortcut method, following commands are identical:

```
my_mc.slideTo(_root._xmouse, _root._ymouse);
my_mc.tween(["_x","_y"],[_root._xmouse, _root._ymouse]);
```

MovieClip.rotateTo()

Availability

Flash Player 6.

Usage

```
my_mc.rotateTo(rotation, seconds, animtype, delay, callback, extra1, extra2)
```

Parameters

rotation A number , end value of movieclip _rotation property

all other parameters do the same as in tween() method

Returns

Nothing.

Description

Shortcut method, following commands are identical:

```
my_mc.rotateTo(180,3,"easeoutback");
my_mc.tween("_rotation",180,3,"easeoutback");
```