

Flash MX – Lesson 3 Movie Target Paths and Loading Movies in Levels. R. Berdan April 12, 2004

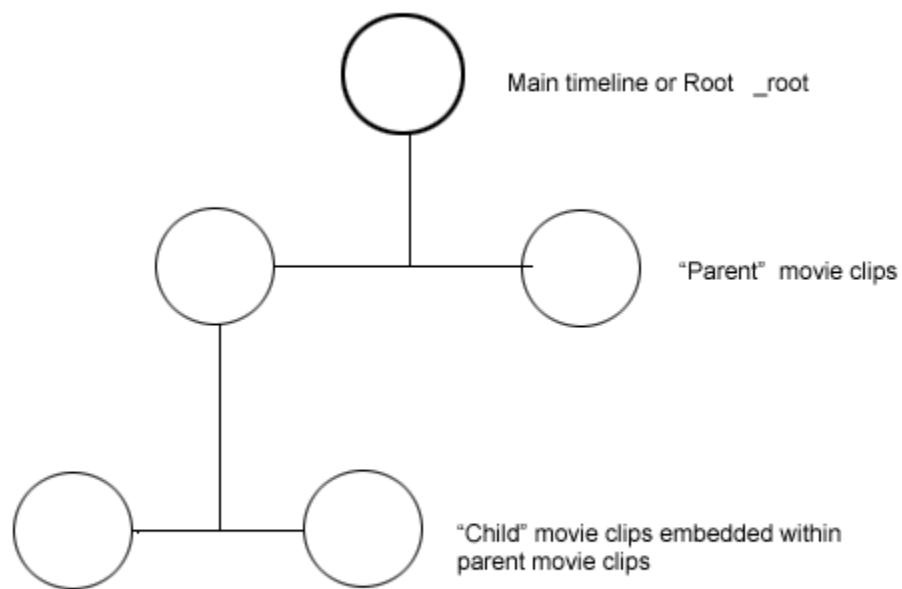
Text book Chapter 3 – Flash MX Action Scripting:

Objectives:

- 1) Understand and use Target paths
- 2) Understand difference between absolute and relative target paths.
- 3) Load external movies and control movies loaded into levels
- 4) Understand how to create global references to access within any movieclip

Introduction:

In Flash it is possible to load several movie clips within another movie clip. This can be done to add interactive components or it can be done to reduce the overall size of the main movie – then load additional movie clips as required.



In order for one movie clip to interact or communicate with another movie clip we need to refer to it by its path.

If you place a button in a specific movie clip in to make the movie play you simply add the following action script:

```
on (play)
{
play();
```

```
}
```

```
or // this.play()
```

If you wanted a button in one movie clip to target another movie clip you would need to add the entire path e.g.

```
on (release)
{
    _root.play()
}
```

the script above would tell the main time line or movie clip to play

```
on (release)
{
    _root.movieclip2.play();
}
```

if the button was in movieclip1, this script would tell movieclip2 to start playing. Child movies embedded within parent movies do not require the full path including `_root`.

TWO TARGET PATHS

1. Absolute include entire path e.g. `_root.movieclipname.action()`;
2. Relative path used to control objects within the same movie e.g. `action()`

TARGET PATHS

Target paths allow you to alter time lines, access variables, data, functions and objects in different movie clips.

Exercise Chapter 3 Text book – Hatfields

Current movie clip: to play a movie clip you simply add the action:

```
play();
```

Or

```
this.play(); // this is a keyword that means current referenced object
```

1) PAGE 89 – Open Lessons 3 – Assets currentTarget1.fla movieclip

The main movie clip has 4 frames with different background colors and a child movie clip.

Add the following script to the first keyframe of the actions layer. :

```
StartingColor = random(4) + 1;  
Trace(StartingColor);  
this.gotoAndStop(this.startingColor) ;  
// not the this keyword is not required here and can be removed
```

Control Test Movie – several times, the background color of the movie will be different as it is randomly selected.

Random() function creates a random number, value of 4 tells it to restrict the random numbers created to values between 0-3, which is why we add + 1 so that we avoid number 0. Try it remove the 1 and watch the trace function output random numbers to the window. The trace function is not required it is used as a diagnostic to display the variable created only and should be removed or commented out of the script once you see what the values are.

2) Double click on the small movie clip inside the main movie clip.

Select the invisible button (button layer – invisible buttons have a light blue color) on the graphic and add this script:

```
on (press)  
{  
startDrag(this);  
gotoAndStop("Speak");  
balloon.text = words;  
}
```

```
on (release)  
{  
stopDrag();  
this.gotoAndStop("Quiet");  
}
```

Speak and Quiet are frame labels, words is a variable which will contain a string of text.

Return to main movie timeline and Test the Script **Control>Test Movie** - you should be able to drag the movie clip around the main movie and when you click on the small movie it should display balloon text box that is empty.

Page 92 – Open the library and drag two more instances of the movie clip onto the main movie and then select Control Test movie – notice that you can drag all of the movie clips. This is because the invisible button is part of the main movie timeline, thus each instance of the

movieclip contains this button as well. (Normally one would edit the movie in the library so the actions would apply to all the movie clip instances).

We can also apply actions in the main time line to each movie clip so they have unique properties.

Close the test window and with the actions panel open, select on movieclip instance and add the following script

```
onClipEvent(load)
{
    words = "My name is Derek";
    this._xscale = 75;
    _yscale = 75;
}
```

IMPORTANT

When we use clip events (load, enterFrame, mouseMove etc), these event handlers affect only a single movie clip instance.

If you wanted to include a script that is shared among all the movieclips you put the script inside the movie clips timeline.

Select another movie clip and add:

```
onClipEvent(load)
{
    words = "My name is Ashlie";
    _x = 400;
    _y = 300;
    // xy coordinates of the movie clip when main movie loads
}
```

and to the other movie

```
onClipEvent(load)
{
    words = "My name is Kathy";
}
onClipEvent(mouseMove)
{
    this._rotation = this._rotation + .5;
}
```

Control Test Movie

Save the file as currentTarget2 fla

TARGETING THE MAIN MOVIE (page 95)

To target the main movie:

```
_root.play();
```

Note movies loaded into levels are also considered to be in the main timeline (next section).

Open rootTarget1.fla in the Lesson 3 Assets folder.



You will see two buttons in the lower right corner – we will add script to these buttons so they cause the main movie clip to increase + or decrease – in size.

Select the minus button and add the following script:

To the - button

```
on (release)
{
    _root._xscale -= 10;
    _root._yscale -= 10;
}
```

Note textbook uses `root._scale = root._xscale – 10;` which is equivalent to the script above

To the + button

```
on (release)
{
    root._xscale += 10;
    root._yscale += 10;
}
```

PS make sure you select the entire button not the + or – symbol on the buttons when you add the script. Control >Test Movie – pressing the buttons will make the movie increase or decrease in size by 10% each time you press the button.

EXTRA This is not in the text book: Notice that pressing the button causes the movie to increase or decrease in increments of 10% - What if you wanted buttons that cause the movie to ZOOM in and out smoothly – How would you do this?

ANSWER

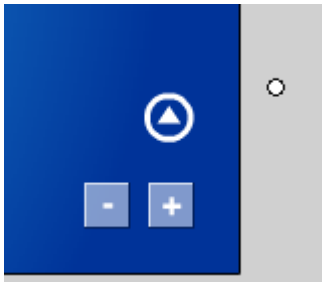
To create a zoom button – you need to add an invisible movie clip and a button. I will show you how to add a + zoom button, you can add the – zoom button by modifying the script.

Insert>New Symbol>Movie clip - if you are taken to movie clip editing return to the main timeline and open the library. (Window>library)



Drag the movieclip onto the stage but outside the main movieclip – see + above to the right of the buttons. Select the empty movie clip and give it the instance Name in the properties box – StoredActions. This is the movieclips name that you will use to refer to it.

Select a button from the common libraries – I selected an arrow button, dragged it on the stage rotated 90 degrees so it pointed up and change the color to white (these are not necessary they just make the button look better).



UP button arrow in a circle., invisible movie clip “StoredActions” on the right white circle

Select the invisible movie clip and double click on it so you are in movieclip edit mode. Add 3 keyframes. (See picture next page). Add the following scripts to each keyframe starting with keyframe 1.

Keyframe 1

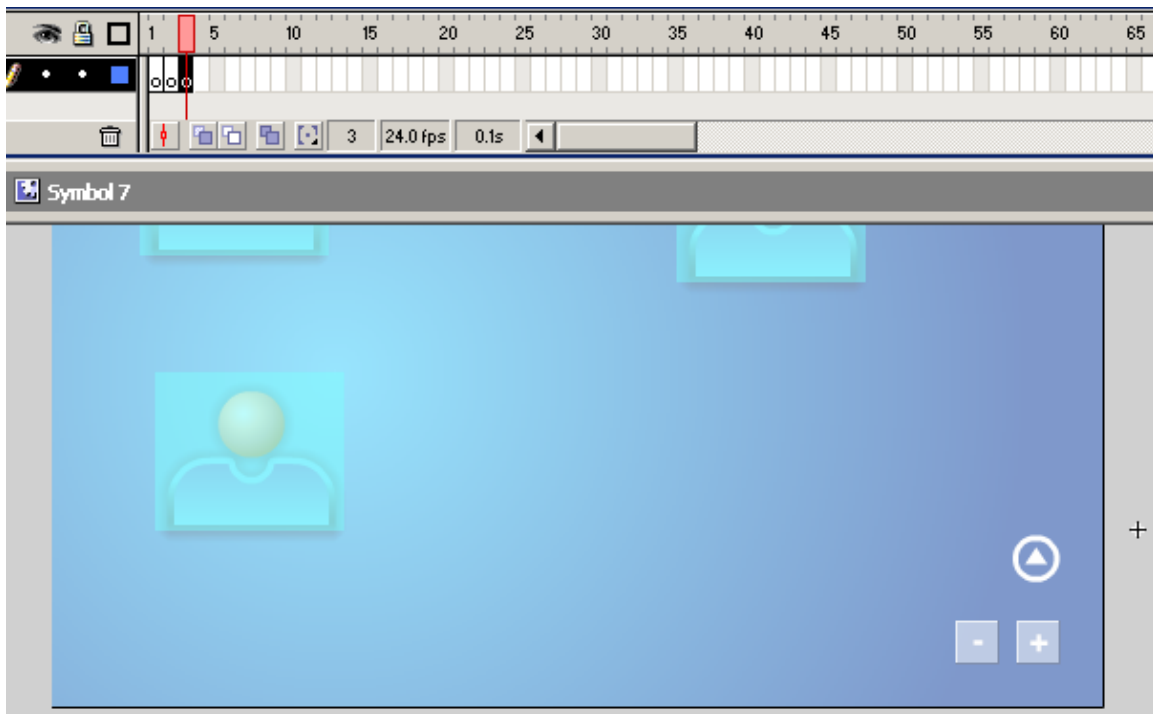
```
stop();
```

Keyframe 2

```
_root._xscale += 10;  
_root._yscale += 10;  
// this causes the _root movie or main timeline to enlarge when
```

Keyframe 3

```
GotoAndPlay(2);  
// this causes the movie to loop continuously when called i.e. zoom
```



Edit mode of the invisible movie clip, add 3 keyframes and add the scripts above.

Return to main movie time line select the invisible movie clip and give it the instance name StoredActions

Select the up arrow button and add the following code:

```
on (press)  
{  
    _root.StoredActions.gotoAndPlay(2);  
}  
on (release, releaseOutside)  
{  
    _root.StoredActions.gotoAndStop(1);  
}
```

Control Test the movie – and it should zoom in.

TARGETING PARENT MOVIES page 97 - Open parentTarget1 fla

Double click on one of the movie clip instances on the stage to edit it in place. With the library panel open and the Child Clip layer selected, drag and instance of the Hatfield Child movie clip onto the stage, just to the right of the graphics in the current movie clip.

You have just placed one movie clip into another movie clip. Notice that all the movie clips have a child movie clips.

Select the child movie clip instance and add this script.

```
onClipEvent (load)
{
    this.words = _parent.words + "'s kid";
}
```

Control Test movie and you click on the child movie clips you should see each indicates the message I am parents kid where parent is substituted by Ashlie or which ever parent movie clip the child is attached to.

SKIP section on page 99

TARGETING MOVIES ON LEVELS PAGE 105.

Using Flash one can load multiple movie clips into a main movie clip. You may want to do this for several reasons. First rather than load all movies e.g. slide show at once, you could selectively load smaller slide shows when required. You can also load different movie clips in order to add interactivity. To do this you load movie clips into levels and then you refer or target the movies by referring to them by their **level**.

To load a movie clip – you first must have the movie clip as a file.swf

In the main movie you call the movie using **the LoadMovie("file.swf", level)** – you can load the movie by placing the script into a keyframe of the main movie or you can create buttons to call the movie clip as you need it.

LoadMovie("url", level/target [, variables])

URL – absolute or relative .swf or JPEG file to be loaded

all .swf files must be in the same folders and filenames can not include folder or disk drive paths

Level – integer greater than 0

Variables – optional, specify method for sending variables as GET or POST

Not movies loaded into other movie clips will inherit the properties of the parent movie clip.

The main movie is always level 0 and this number can not be assigned to any other movie clip, otherwise you can load a movie into any level 1,2, 247 etc with no other restriction. Once loaded you can target the movie clip by referring to its level and attaching an appropriate action.

Eg.

```
_level37, play(); // will cause the movie clip on level 37 to play
```

Targeting movies on different levels

First Note that you can specify a target or a level – BUT NOT BOTH!

Movies on the same level can be targeted using a relative path

e.g.

```
on (press)
{
_level1.alpha = 50;
startDrag();
}
```

FOLLOW instructions starting on Page 105 in the text book – you will first prepare two external movie clips which you will make invisible, then export them. In the main movie you will load the movie clips in the first keyframe (but they will be invisible) they you will make them visible, resize them and send text from the main movie to one of the movie clips.

GLOBAL ELEMENTS (page 118).

If you want to be able to reference or call a variable or function from any movie timeline without including the absolute file path you can turn the variable or function into a global element.

E.g.

```
_global.myVariable = "hello";
```

can reference the variable from any movie clip

```
_root.mymovie.greeting = myVariable;
_root.greeting = myVariable;
```

Functions

```
_global.myFunction = function()
{
// statements;
}
Call from any timeline
MyFunction();
```

It is recommended that you precede Global variables with a small “g” e.g. gmyVariable to prevent naming collisions in your scripts.