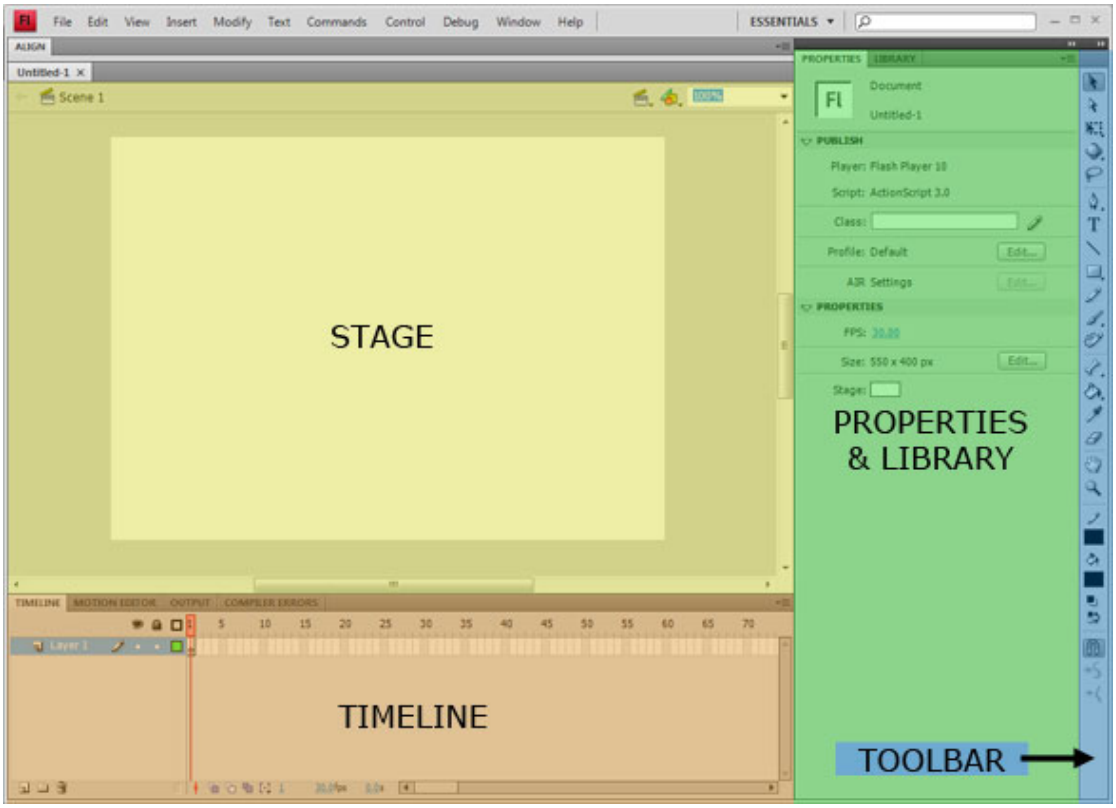
Introduction to Flash

This week, you will be introduced to FLASH animation 101. Along with simple animation we will be learning how to create buttons.

Flash Interface

Welcome to Flash. Flash is an incredibly powerful program that has seemingly endless potential. Flash can be used for creating games, making presentations, animations, visualizations, webpage components, and many other interactive applications. Some of the Flash interface components will look familiar to you, as they have the same functionality as other Adobe applications. However, Flash requires a certain mindset to work in it properly, especially when animating with vector graphics and coding with actionscript 3.0.

Here is an overview of the Flash interface.



Stage: The stage is the main workspace of Flash, all your compositional elements (movie clips, buttons, graphics, and etc.) will be arranged here.. Content that is within the box in the middle of the stage will be visible when the Flash movie is output. The grey background area outside the box in the middle is 'off-stage'. You can animate content from off-stage onto the main stage area or use a background image that is larger than the main stage to move around as if the camera is panning across a background. The Stage has several context which are indicated along the top bar of the stage. It can present content that is in a Scene or can present sub-content such as objects from the library. You can show and hide the things that exceed the size of your stage by turning off/on Pasteboard (View>>Pasteboard)

Timeline: The numbers across the bottom correspond to the frames that occur as time progresses through the movie. You can navigate to any frame of your animation to perform editing. Also, Flash has layers just like many other Adobe applications. These appear along the left side of the Timeline.

Properties: The Properties tab changes depending on which tool on the toolbar you have selected or which object you have selected on the stage. Each object or tool has its own properties which can be adjusted in this tab.

When you have the Selection Tool selected and click the background of the stage, the Properties tab shows the Document Properties. Here you can set the size of your Flash file, background color, frame rate and exporting settings.

Library: The library tab will be your best friend in Flash. It holds all the symbol objects of each Flash file. You can organize your library like you did in your windows/mac directory. (i.e. created new folders for different types of elements, or nest one symbol in another)

Drawings & Symbols

There are two primary types in Flash, Drawings and Symbols. Drawings are created with the vector editing tools in Flash such as the Pencil, Brush, Oval, Rectangle, etc. Drawings have stroke and fill which can be reshaped via the Selection and Subselection tools as well as any other vector editing tools.

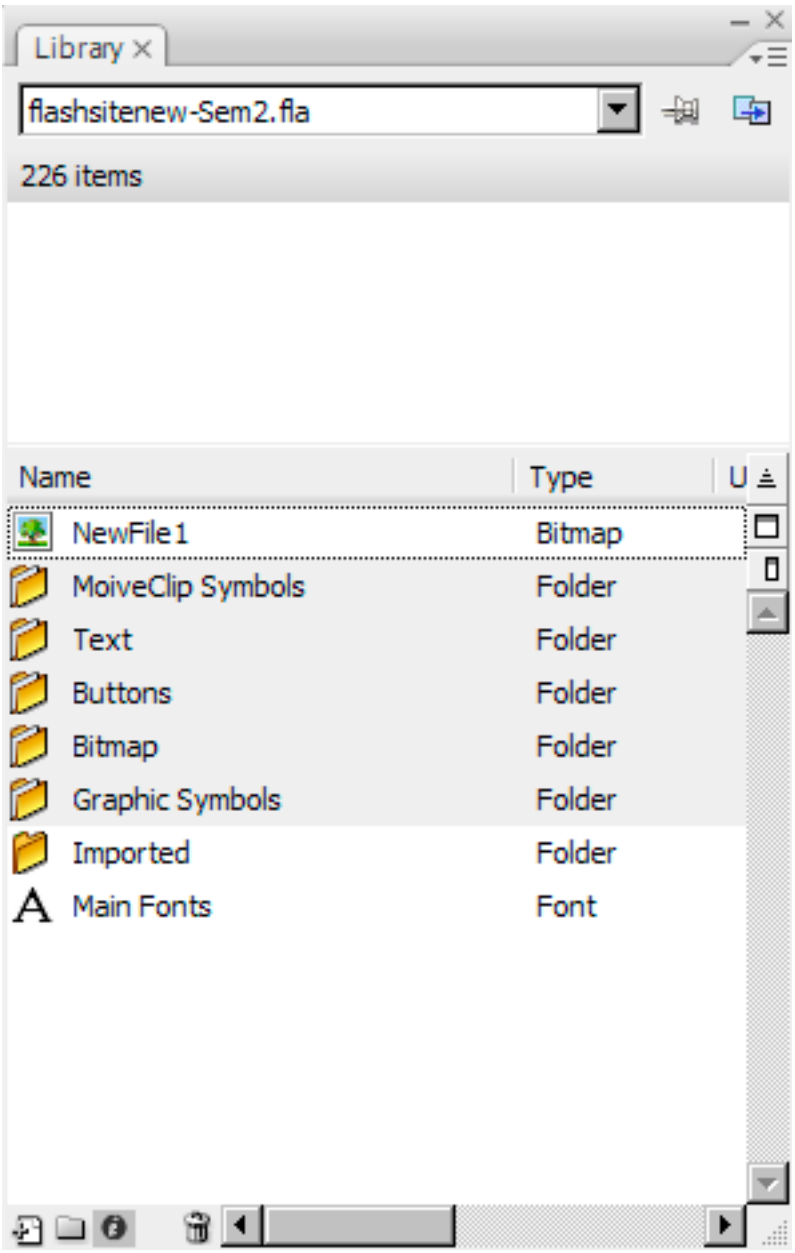
Symbols are a powerful feature of Flash in that you can draw an object once, save it in your library, and use it over and over. Symbols can be created from a Drawing, collections of Drawings and other Symbols, or external files (JPGs, PNGs for instance). Basically, symbols are like container for your visual/auditory elements. Instead of having hundreds of graphics, text, and other elements laying on the stage, you can create a symbol and use wherever you need it. If you want to change the appearance of every instance of that object throughout your movie you can directly edit the Symbol in the library. You can also break apart Symbols into their original objects to make derivatives of a Symbol.

Symbols: Movie Clips vs Graphic vs Button

There are three types of Symbols: a Movie Clip, a Graphic symbol, or a Button symbol.
These symbols can have actions applied to them to dynamically load, disappear, trigger other events, and many other things (in actionscript). The fundamental distinction between Movie Clips and Graphic are that the former has it own independent timeline whereas the latter shares the same timeline with the stage's timeline. Button symbols are simplified objects with special frames for a mouseover appearance, and a click appearance.

Library Structure

Once you start to work on a flash project, most likely you will have a lots different types of files(text, graphics, video, imported stock images), you want to have your library organized in a way that files of the same type should be placed together and give your folder recognizable names so when you need to find some elements you know where to look for it. It will save your time and hassle, also it will make your work flow better. This is very critical if you are working in a team, because when you pass the file you have worked on to other people; if the library is not well-structuralized; your co-workers will have hard time to simply start it.

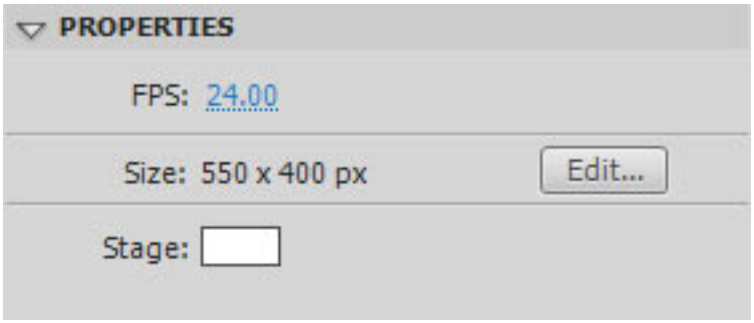


Flash Project Properties

When you first create a new Flash project you are presented with the default properties page on the right side of the screen. These properties all have a significant impact on the movie and should only be adjusted at the beginning of a project or a significant amount of work may be required to correct the issues that arise.

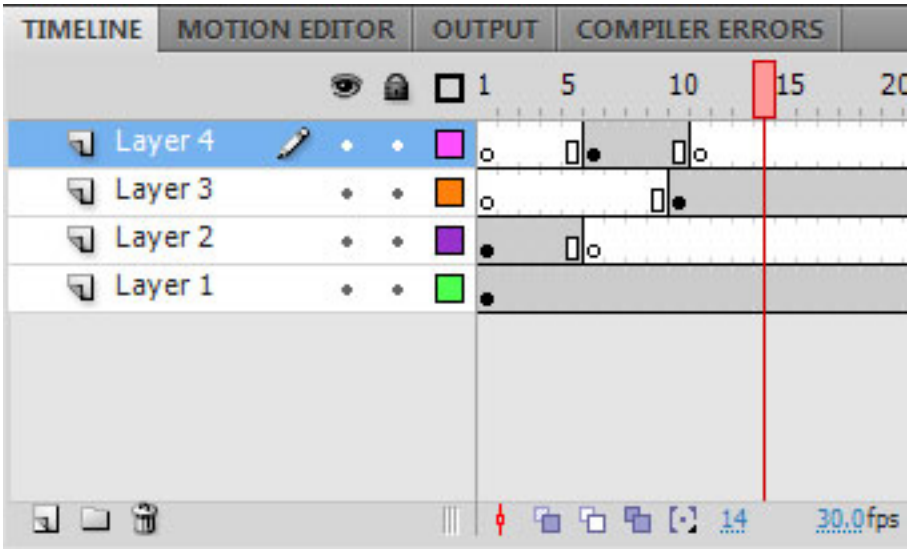
FPS stands for Frames Per Second and is the frame rate of your movie which determines the smoothness of your animation. For fluid motion the frame rate should be left at 24-30 fps. If your animation doesn't have much movement or doesn't require this level of quality, reducing the frame rate can significantly the file size of your resulting movie. Note that while changing the frame rate does have an overall impact on the speed at which your frames play, you should never use the frame rate as a means to adjust the speed of your movie.

This property page also allows the adjustment of the size and color of the stage. The stage determines the output size of the movie, while the size can be adjusted in the player, you achieve your best quality setting your stage to the maximum size you expect for your final output.



Timeline & Frames

In Flash we place all content in layers just like in Photoshop. The same tools exist to create a new layer, delete a layer, and create folders for layers. The primary difference is that content must be placed into a Keyframe on the timeline. Keyframes are frames in the timeline that contain information (content or script). Keyframes without content are denoted by an empty circle, keyframes with content have a filled circle. You can right-click on the timeline for a specific layer and select 'Insert Blank Keyframe' to begin adding content (when you create a new layer there is always a blank keyframe at frame 1). Click a blank or empty keyframe in the timeline to add content to that frame. A keyframe only lasts for one frame (if your movie is 30 frames per second a single keyframe is 1/30 of a second). To extend the length of a keyframe, right-click onto a frame further down the timeline and select 'Insert Frame'. You can also select, drag, copy and paste frames. Note that in order to perform any operation on a frame you must select it first.



Flash Animation

Animating in Flash requires certain understanding of motion and a lot of patience. Figuring out how the animation will work is not so hard, but achieving the desirable result is usually painstaking. In the tutorial,

we'll explain the timeline and frames, tweens, and other basic techniques needed to make animations happen. You'll be animating things before you know it.

Pose-to-pose animation in Flash is created by defining actions in-between two points on the timeline, hence the term 'Tween'. There are two types of Tweens in Flash, Shape Tweens, which can only operate on drawing objects, and Motion Tweens, which can only operate on Symbols. Shape tweens allow you to mutate from one drawing object to another drawing object with different properties or at a different location on the stage. Motion Tweens allow you to move an object along a path, resize or change some basic properties of a symbol.

Shape Tweening

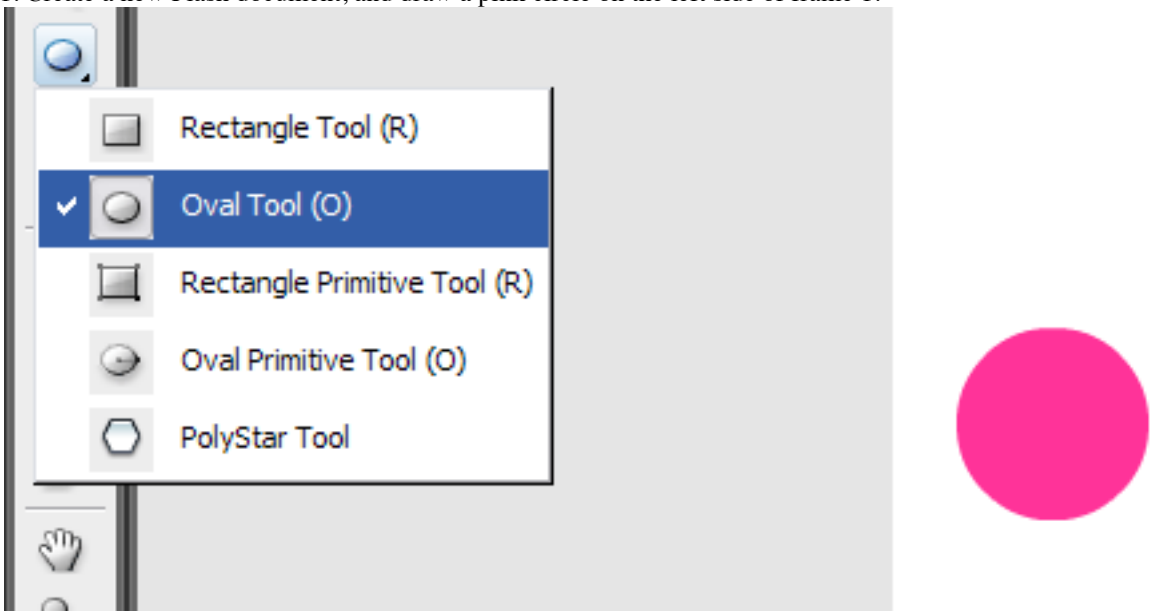
The basic idea is that at point A in time you have one object, and at a later point B you have another object. Between the two points, you have a gradual shapeshifting transformation from object A to object B. In order to Shape Tween, you need two keyframes to mutate between. Like this, for example:

(Note that to tween text in Flash requires breaking apart the letter symbol)

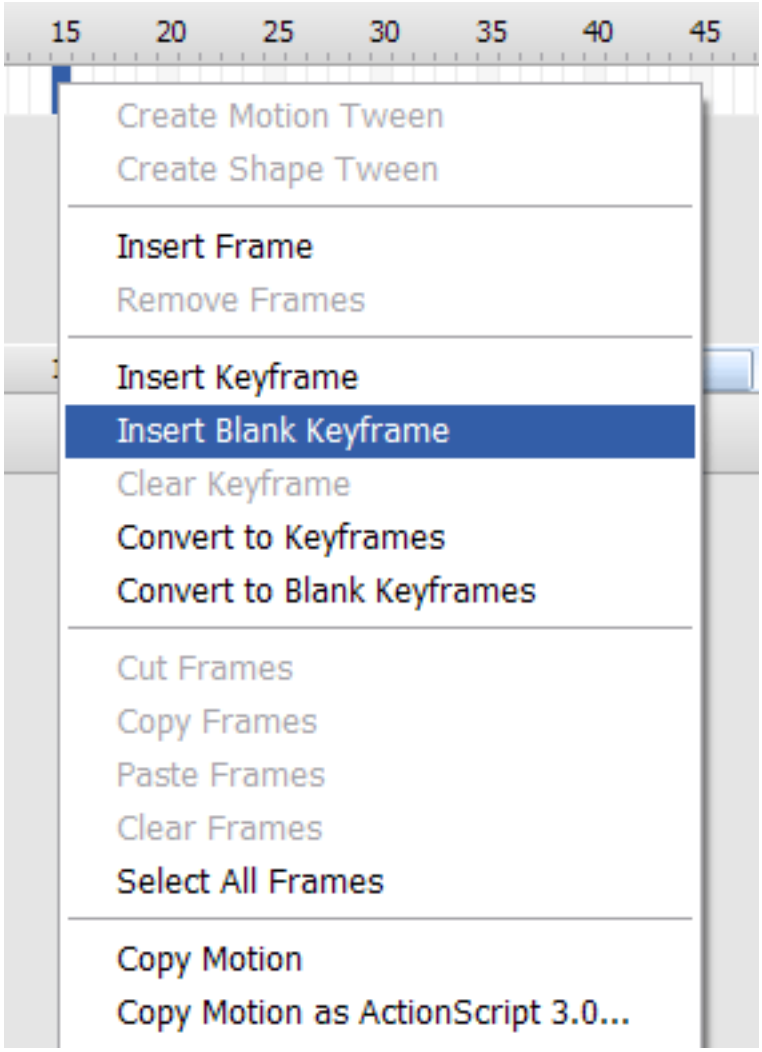
Lets get started on our first tutorial.

Pink Circle to Blue Square

1. Create a new Flash document, and draw a pink circle on the left side of frame 1.

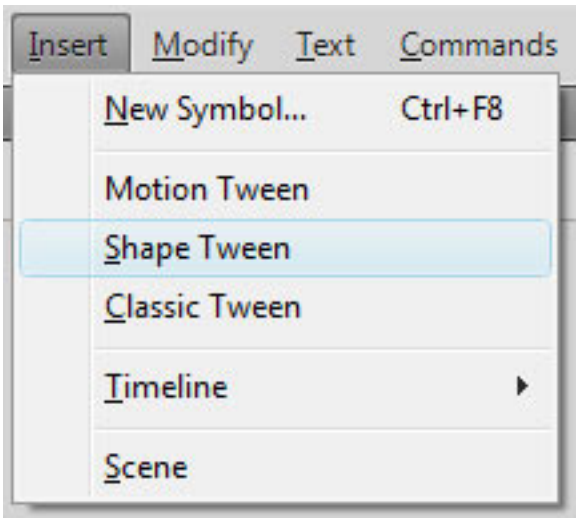


2. Click frame 15, and right click to insert a blank keyframe. You now have 15 frames full of nothing but a circle.



3. Select the keyframe at frame 15, choose the rectangle tool (not the rectangle primitive tool), and draw a large, blue square on the right side of the work area.

4. Then select anywhere in the extended frame 1, and Insert a Shape Tween from the menu (or rt-click on the frame and select Shape Tween).



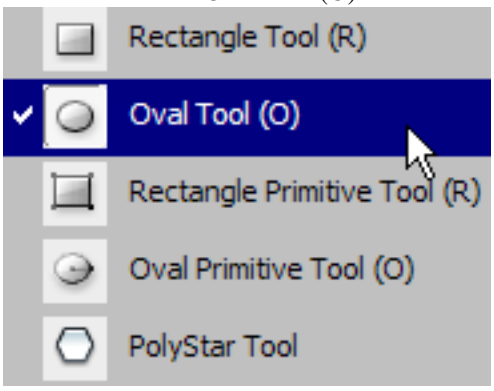
What we've just made is a shape tween. Between frames 1 to 15, a transformation occurs.
The most important thing to remember when creating shape tweens is that unlike motion tweens, shape tweens must involve shapes or drawing objects and not groups or symbols. For a shape tween to work, the basic attributes "the stroke and fill" must be able to change so that it can morph the original shape into something else. The simplest way to ensure that all the elements you want to shape tween are “shape tweenable" is to select all the objects and use the Modify > Break Apart menu option to ensure they’re broken down into their constituent elements.
To view what you've just made, CTRL- ENTER.

Motion Tween

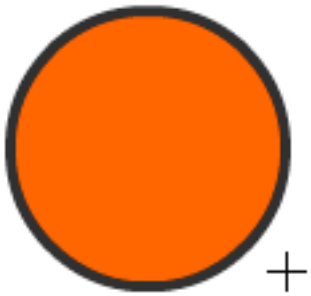
Motion tweens tends to produce smaller file sizes, and they tend to be easier to produce. If you have a choice between using either shape or motion tweens, it's generally better to go for motion tweens, and use shape tweens as sparingly as you can. Having said that, shape tweens give the experienced animator more control. Motion tweens allow you to make the subtle animated changes that add emotion and expression to characters, such as changes in facial expression during speech, or subtle body movements during walking.

Drawing with Shape Tools

1. Let us select the Oval Tool (O) from tool bar and drag-draw a circle;

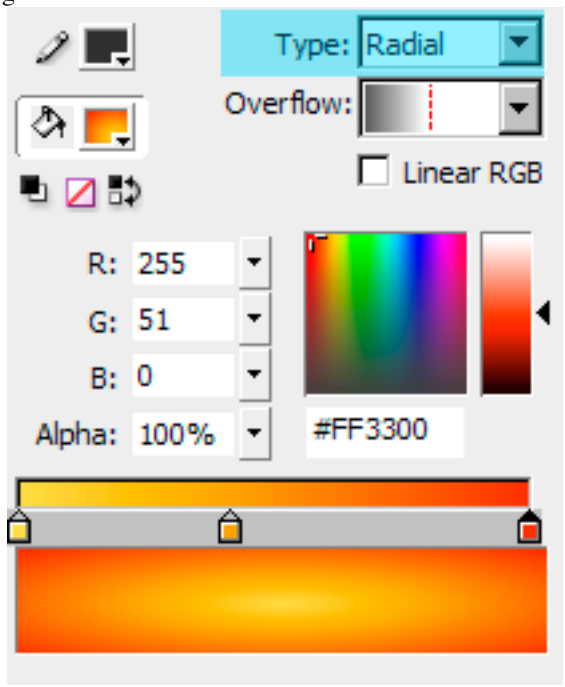


2. Go to Fill Color and change the color to vibrant orange; Also, go to property inspector and disable the stroke contour;

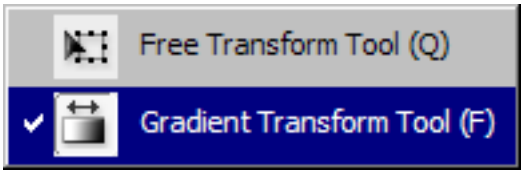


Applying Gradient

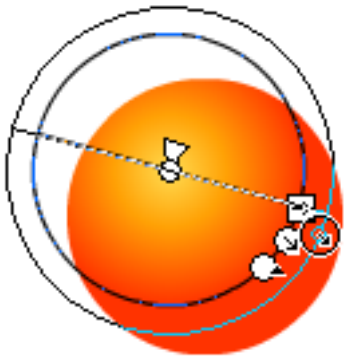
3. Go to the Color Panel, select Radial under the fill type(highlighted), and add an new slider in the gradient control so we can create a more smooth and naturally looking gradient.



4. Go to Tool Bar and Select Gradient Transform Tool;

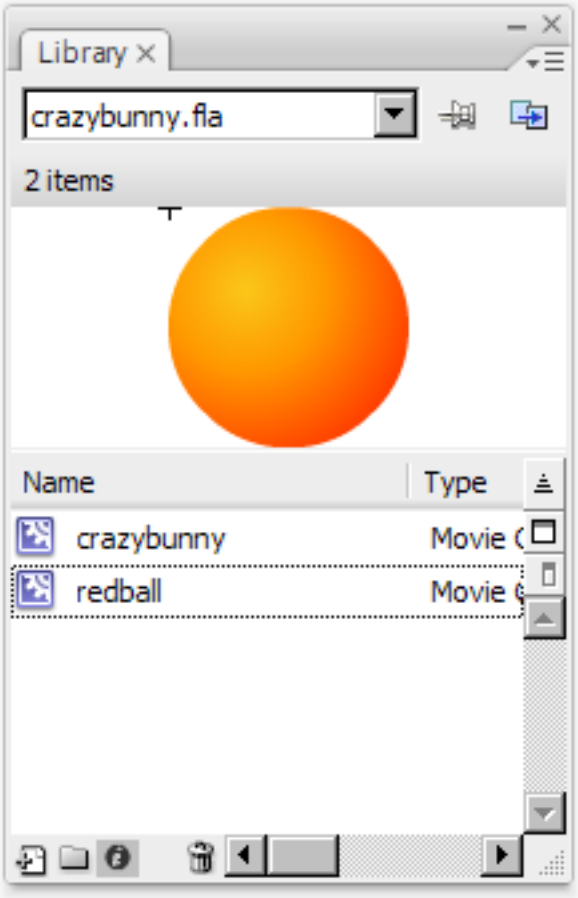


5. Click on the orange circle; now you can control the size, direction, and center point of the gradient.



Converting to Symbols

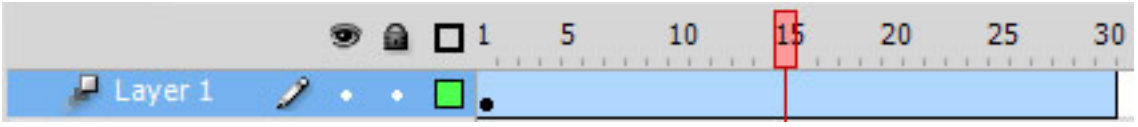
6. After you are happy with what you got, click on the ball and hit F8 or Right-Click >>Convert to Symbol; set the symbol time to movie clip; note that the symbol you've just created is now in the library; then delete everything in the stage and drag the movie clip: redball to the stage in frame 1.



Adding the Motion Tween

7. Now we are going to animate the ball and make it bounce up and down. Click on frame 30, right-click and select 'Insert Frame' to extend the frame for 30 frames. Next right-click anywhere in the long frame that was just created and select 'Create Motion Tween'. Note that when we create the motion tween the long

frame turns blue and the rectangle at the end disappears. The layer also now has a Tween icon next to its name.

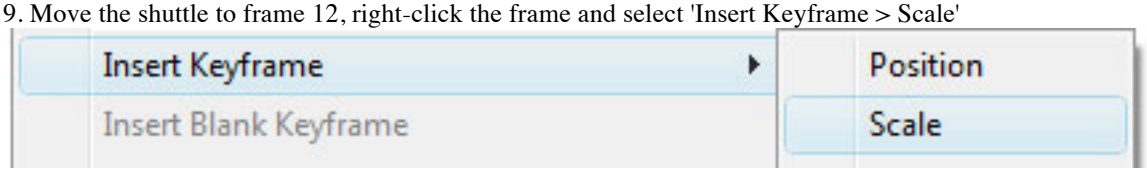


8. Now we need to add some movement to the tween by placing the ball at different places along the timeline. In a Motion Tween you move the small red rectangle at the top of the timeline, the shuttle, to the location where you want to change tween properties. Let's move the shuttle to frame 15. Next hold down the Shift key and click and drag the ball to the bottom of the visible stage (holding the shift key ensures that we will drag the ball in a straight line). Next move the shuttle to frame 30 and shift-drag the ball back to the top of the screen. Notice the green motion path created where the ball will travel.

Run the animation by pressing CTRL-ENTER.

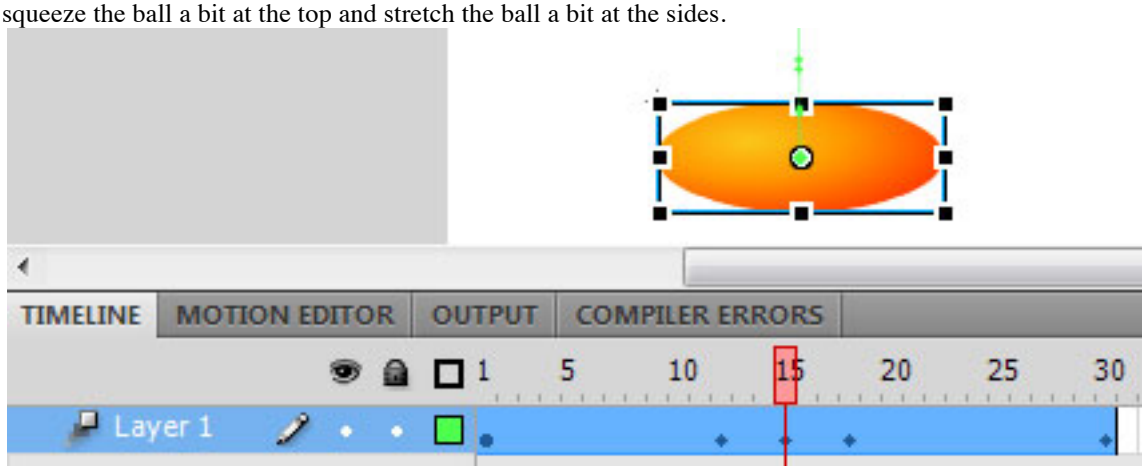
The animation works, however, it lacks a realistic feeling since it moves up and down constantly at the same speed; what we want to do now is to add some real-life ingredients; when a ball hits the ground in real life, it will squeeze and squash.

9. Move the shuttle to frame 12, right-click the frame and select 'Insert Keyframe > Scale'



This adds a small black dot to the frame and saves the scale of the symbol at that frame. Do the same thing for frame 18.

10. Finally, move the shuttle to frame 15 again, select the Free Transform Tool from the toolbar and squeeze the ball a bit at the top and stretch the ball a bit at the sides.

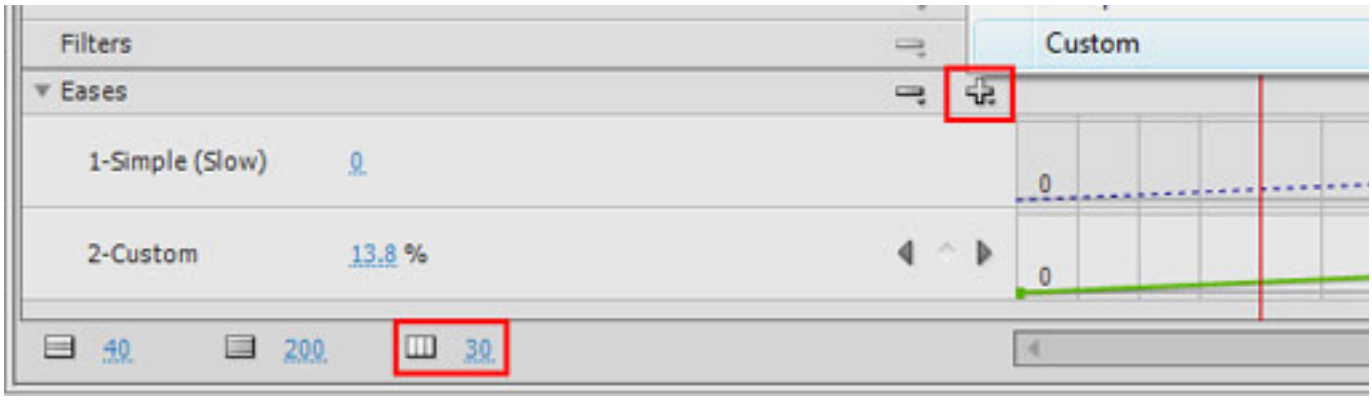


Hit CTRL-Enter to see the animation with the squash and stretch.

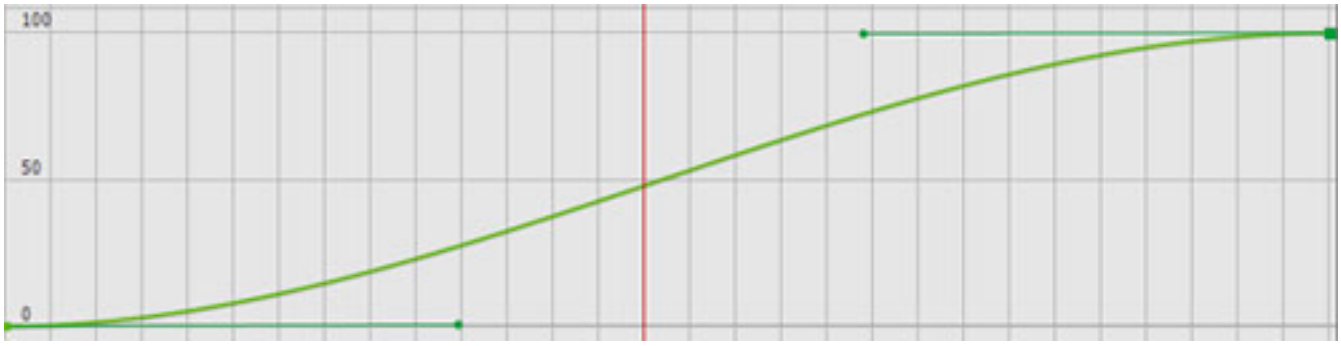
Easing In/out

11. Gravity speeds things up when they are falling, if you need more touches of real-world physics, you can use Easing In/Out to make your object move more naturally. Simple easing can be done by clicking on the tween and adjusting the Ease in the property tab. If we had seperated this into two tweens, we could have set one to ease in and one to ease out. However, we want one tween to both ease in and ease out. Custom easing in Flash CS4 is handled through the Motion Editor. Select the Motion Editor tab next to the

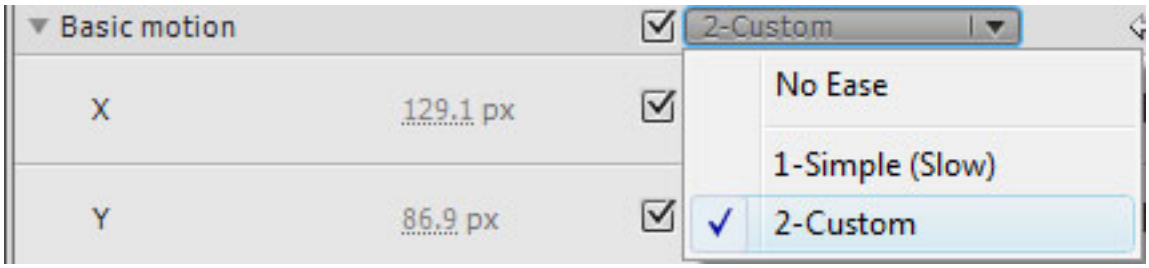
Timeline tab. Scroll down to the secion labeled 'Eases', click the + sign and select Custom (the red box in upper right of the diagram below). Make sure that the scale indicates 30 (the red box in the lower left below).



Click where is says '2-Custom' to expand the graph. Click the left control point, grab the handlebar and move it along the x-axis. Then click the right control point, grab its handlebar and move it so it is along the x-axis like below.



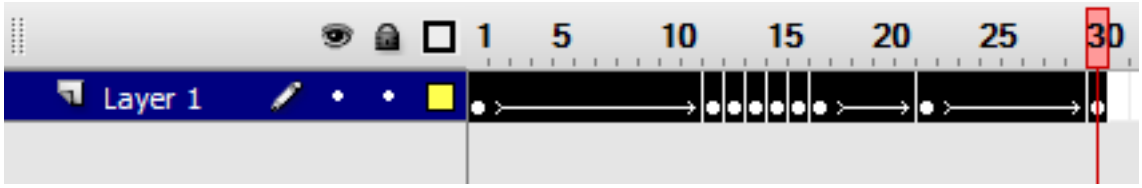
Finally, scroll up in the Motion Editor to where it says 'Basic Motion' and in the drop-down select '2-Custom' to indicate that we want our custom ease applied to the motion.



Here is the final result of a simple bouncing ball animation.

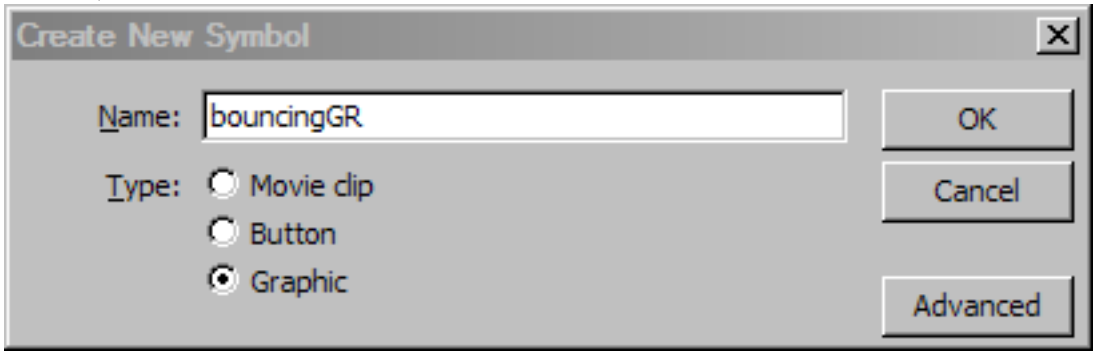
Nesting Animation

Nesting means to place all the compositional element in one of your animation sequence into a single symbol; It is an incredibly powerful technique for animators.
With nesting, we can easily create animation like this without using motion path;
1. Let me show you how it works in this tutorial; first, let us select all the frames on the timeline of your bouncing ball animation;

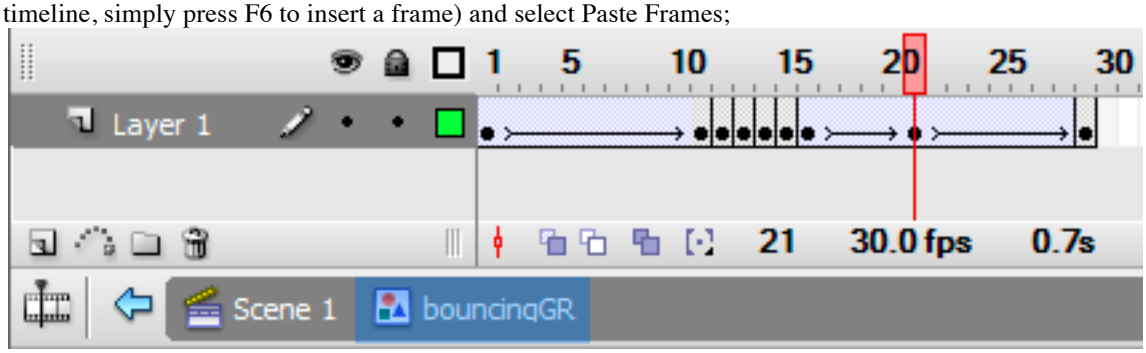


2. Right-click and select Copy Frames;

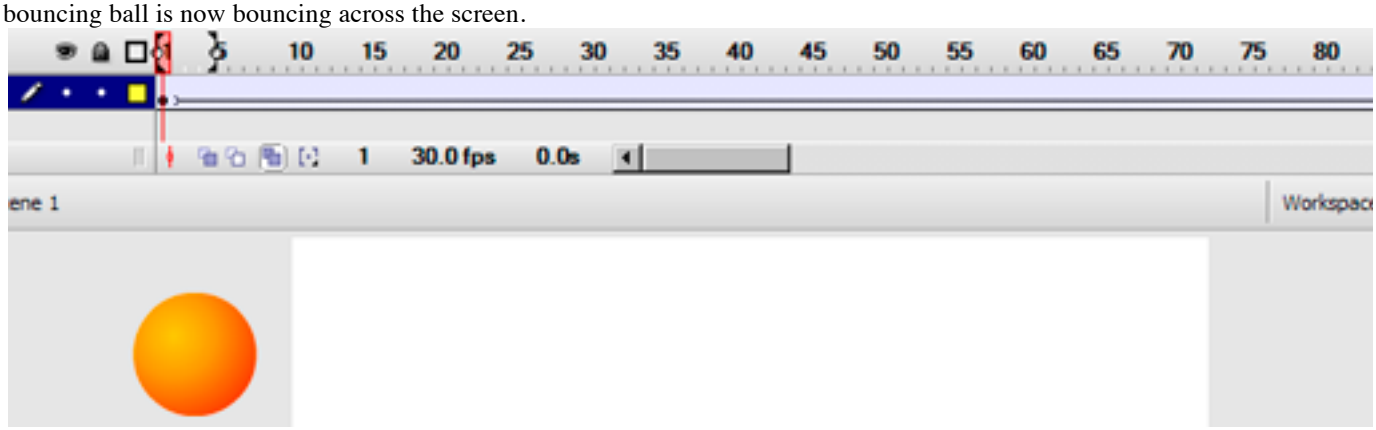
3. Create a new symbol named bouncingGR(you can use anything you want), select Graphic as its type and click OK;



4. Double click on the new symbol in the Library, right-click on the timeline(if there is no frame on timeline, simply press F6 to insert a frame) and select Paste Frames;



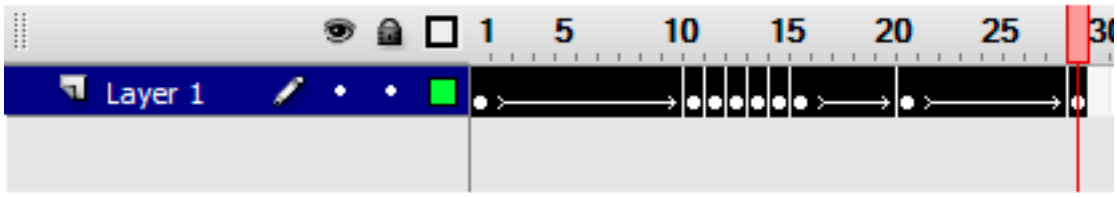
5. Now we can delete everything on the scene1's timeline since the animation has been pasted into a symbol; then let us drag the symbol bouncingGR to the stage; On frame 1, move the symbol all the way to the left of the canvas; next, click on frame 90, press F6 then move the symbol all the way from the left to the right of the stage; finally, let us create a motion tween between this two frames, then you will see the bouncing ball is now bouncing across the screen.



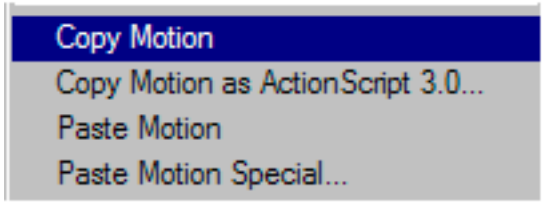
Copy Motion

In Flash CS3, not only that we can copy frames, we can also copy motions. Copy motions allows you to apply the same motion to a complete different object.

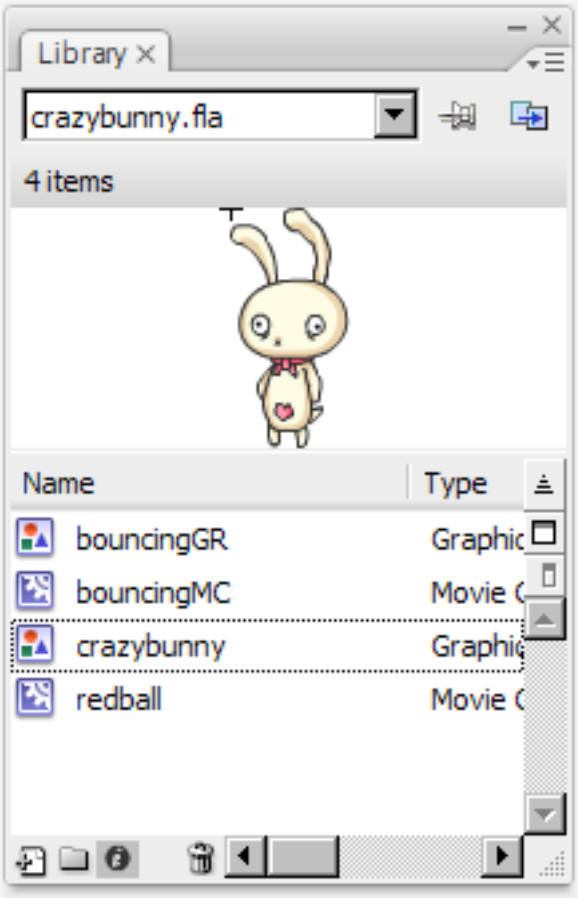
(1) Select all the frames



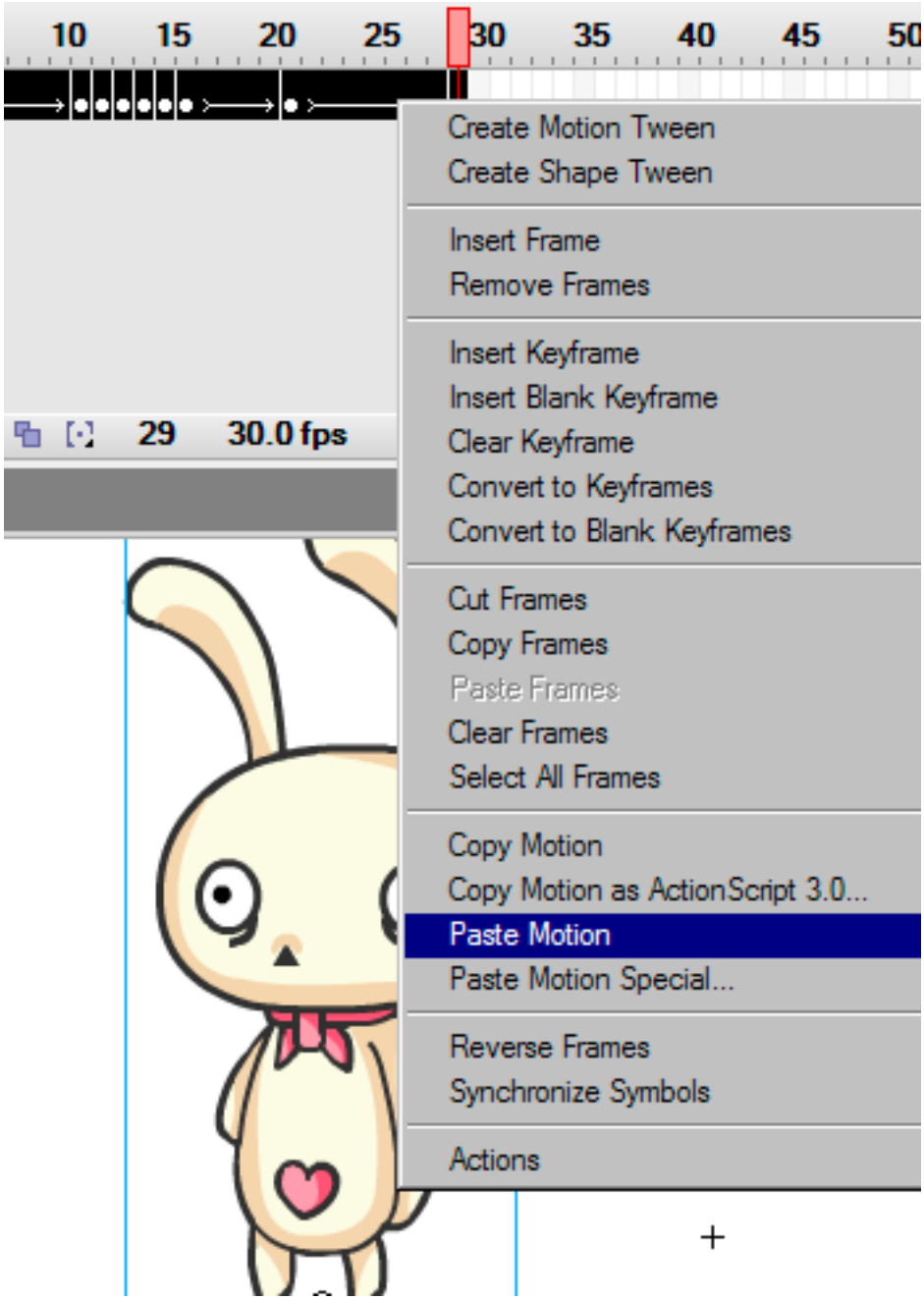
(2) Right-clicked on the timeline and Copy Motion



3. Drag the crazybunny graphic symbol to the stage(or if you have not imported it yet, just import the crazybunny.swf file and convert to a symbol) ;



4. Select on the bunny, click on the timeline and right-click>>Paste Motion;



6. Now this bunny is doing exactly the same thing as the bouncing ball you just animated; Copy motion is a very simple technique but it is a huge time saver.

Motion Path

Note that the green line which appeared when we first moved our symbol after creating the motion tween is its motion path. The motion path is editable like any line. Click either end of the line to begin editing (both should be in the same place with our ball) and now you can drag the control points to make the ball move anywhere on the screen.

Flash Masking

Here we are going to create a few animations using motion tweens in Flash. The finished file is in the source file masking.fla, but we will show you how to create this from scratch.

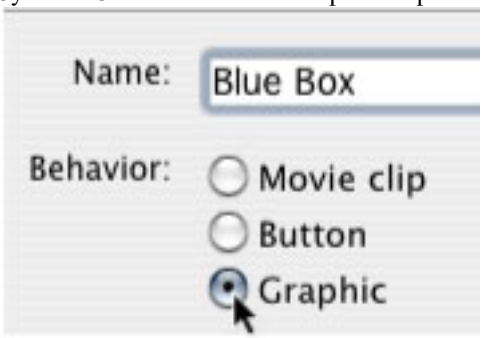
1. This is the animation we are going to be making. Now lets make it.

2. Before we create this animation, its good to know that usually you should be working in 30 Frames per Second or higher. Flash's default frame rate is 12, but you can set it by choosing the selection tool (v) and looking at the contextual tool bar at the bottom of the screen. For this animation we will be working at 12 FPS, but usually you will want to change this, it will allow you to make smoother and cleaner animations.

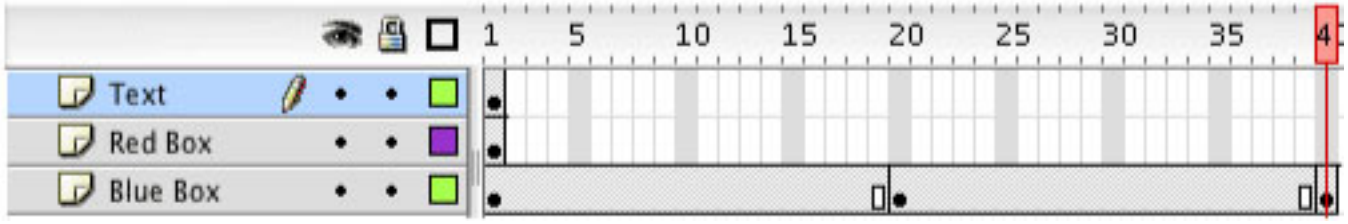
3. Next we are going to draw the first thing we want to animate, which is a box. Click on the Rectangle tool and draw a box on the stage.

4.Since we are going to animate 3 things we need to make 3 layers. You can create a new layer by clicking the Insert Layer button at the bottom left corner of the score.

5.Now we are going to turn the object that we created in Layer 1 into a Symbol so we can animate it. You do this by selecting the shape and pressing F8 or by going to your menu and choosing Modify > Convert to Symbol. Once the box comes up with options in it, name your symbol and choose the behavior: Graphic.

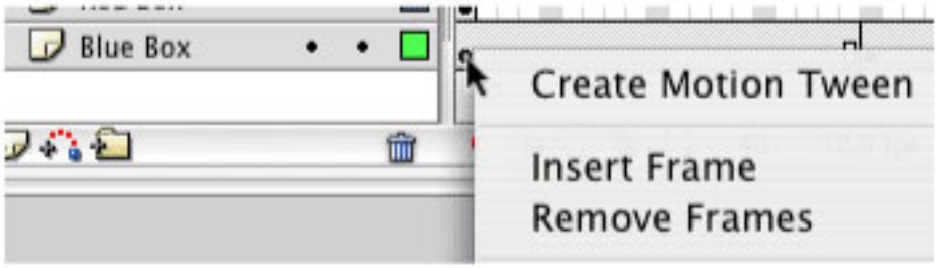


6.You can now create the keyframes for your animation. The way this method of animation works (tweening) is by you creating keyframes (the frames with the little black dots on them) and on those frames setting the important positions of your shape. Then when you apply a motion tween, flash will fill in the frames between the keyframes for you. So the first keyframe, we have the box at the left side of the stage. To make a second keyframe at frame 20, click on the frame and press F6 to create a new keyframe. Then click on frame 40 to create another keyframe. Since we want the box to go back and forth we are going to want the middle keyframe to have to box at the right side of the stage. So click on the keyframe and move the box to the right side of the screen. Now your 3 keyframes should have the box on the left, then on the right, and the final one on the left again.

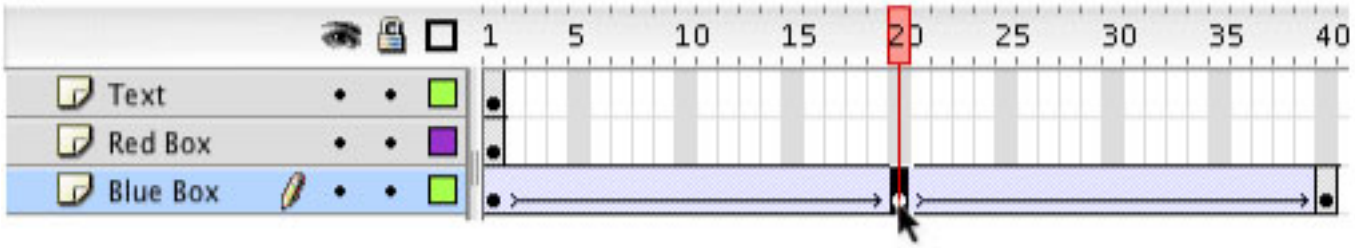


7. Now we want flash to fill in the frames between our keyframes so that we get an animation. (1) Right click on each keyframe in the Blue Box layer and choose: Create Motion Tween. If you have done everything right the frames should turn blue with a little arrow (2).

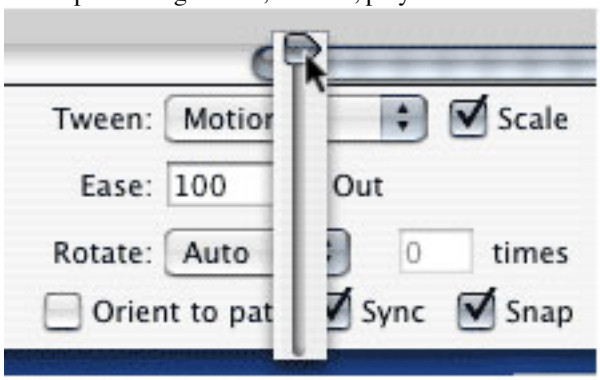
(1)



(2)

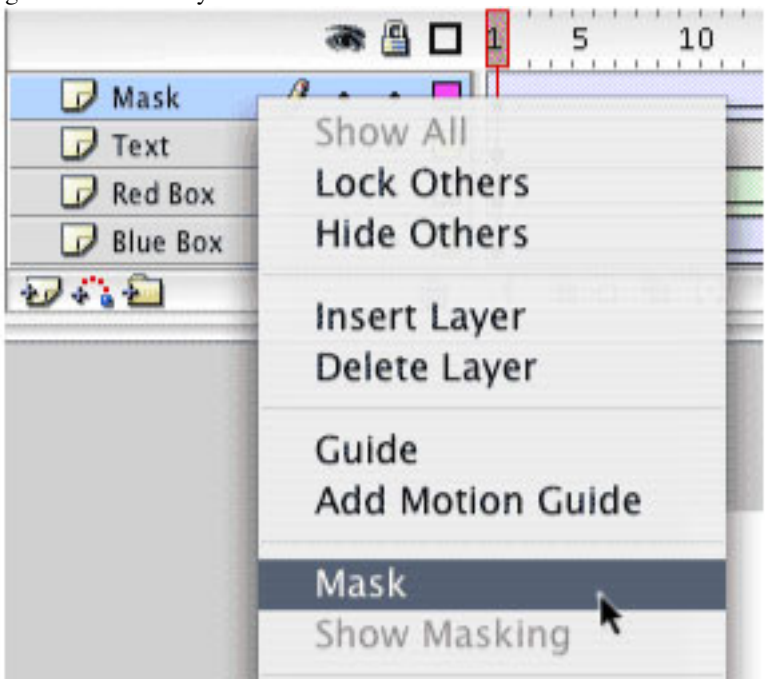


8. If you drag through the score, or press RETURN, your animation should play through now. The box you created should be sliding back and forth. To create a more smooth and organic looking animation click on the keyframes that you have applied a motion tween to, and in the contextual menu at the bottom of your screen put Easing to 100, or -100, play around with the effects of this.



9.Now in the layer above your blue box layer, draw a circle. We are going to create a Shape Tween animation with this shape. You need to make 2 more keyframes using F6 just like you did for the Blue Box animation, however, in the middle frame, delete the circle and draw a different shape somewhere else on the stage. In this case I made a box. Now, instead of right clicking on the keyframes and selecting motion tween, you need to click on the keyframes and go down to your contextual toolbar where you can choose Motion from the Tween pull down menu (1). If you did everything right your shape tweened frames should be green with little arrows in the frames, as you can see in figure (2).

10.So now we are going to make 3rd and final animation, and that will be animating a mask over some text. The first thing you want to do is create your text using the text tool (t) in a new layer. Fill in 40 frames with this text, so click on frame 40 and press F5, this will create frames in the score. Now that you have your text, create a motion tween animation on a layer above it just like you did for the blue box. Make sure your animation is happening right over the text. Now all you have to do to turn your animation into a mask is right click on the layer and choose: Mask.



11.To preview your animation in the end just hit Command-Enter or Control-Enter on a PC. And that's the finished animation. If you have any questions or you are confused about something your mentors in class.

Masking Exercise

Now you know the simple mechanic of masking, Give this a try! (source files are in the Masking.fla library)

Flash Tips - Pan, Zoom, Fade

Some of the best results in Flash can be achieved through the simple techniques of Panning, Zooming, and Fading. Examples have been created in the **flash_panzoomfade.zip** file in the Downloads section on the right.

Panning

Panning is moving the camera across a wide background and stopping at significant spots where the action is taking place. To pan scenes in Flash, simply create a background image that is significantly larger than the stage. Create a motion tween which moves the image across the visible portion of the stage.

Zooming

Use zooming to bring the camera close to a single object or to widen the visible range to include a larger amount of the content. Zooming in Flash is achieved by using a motion tween and the Free Transform Tool (while holding shift to constrain proportions) and just resizing elements on each layer. Select the frame where you want to start your zoom, right-click the frame and add a Motion Tween. Note that when you are zooming multiple layers you will need to add a motion tween to all layers that you are zooming. Next move the shuttle to the frame at which you want to start the zoom, right-click in each layer at that frame and Insert Keyframe > Scale (so that it doesn't start zooming prior to that frame). Move the shuttle to the frame where you want your maximum zoom, and select the motion tweens in each layer to be zoomed by holding down the shift key and clicking each tween. Finally, select the Free Transform Tool, hold shift, and increase or decrease the size of the content as needed.

Combine pan and zoom to great effect.

Fades

To fade in or out, simply create a new layer, insert a Blank Keyframe at the location you want to start your fade, and draw a large black rectangle over the entire visible portion of your stage. Right-click the rectangle and Convert to Symbol and name it Fade. You can now use this symbol any time you want to fade in or out in the future. Next, right-click the frame where you want your fade to end and select Insert Frame to extend the black rectangle until that frame. Add a motion tween to the black rectangle frame. Now, you merely change the alpha to 0% or 100% at either end of the tween depending on whether you are fading in or out. To fade in, move the shuttle to the last frame of the tween. In the properties tab under 'Color Effect' select 'Alpha' from the drop down and move the slider to 0%. Voila, a nice clean fade in.

Action Script 3.0

Introduction

Aside from making animation, Flash is also well-known for creating interactive applications. Action Script 3.0 in Flash allows you to create all kinds of fully interactive applications such as dynamic websites and computer games. Also, AS3.0 is an object-oriented programming language; if you are familiar with AS3.0, it will help you to learn other object-oriented language such as Javascript.

In this sort lesson, our goal is to provide you with some basic understanding of some crucial concepts and the syntax; so once you can find ready-made AS3.0 online and modify it to suit your own purpose. Unfortunately, if you had previous experience in AS2.0, the chances are that it will not help you learning AS3.0 as much and it might even slow you down. The reason is that in AS3.0, all the coding methods are standardized, therefore it becomes less visualized. It would be easier for programmer, but it become harder for designer since you do not immediately see you have done with your code.

Graphic VS Movie Clip

If we want to use AS3.0 to interact with your symbols/objects, we need to set the type of your symbol to Movie Clip; the fundamental difference between Graphic and Movie Clip Symbols is that graphic symbol animates with timeline whereas Movie Clip symbol animates independent of timeline.
Here is a flash movie shows you the difference between Graphic and Movie Clips; you can click on STOP to stop the timeline from playing as well click on PLAY to resume the playback on the timeline.

Instances and Naming Convention

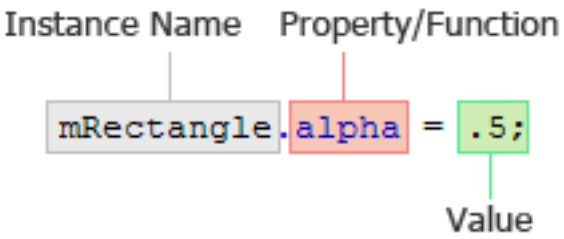
As I mentioned before, in order to access a stage element, we need to give it an instance name; you might find we already give it a name when we convert it to a movie clip. However, that name cannot be referenced by AS3.0 at all. So under the properties inspector (keep the onstage movie clip selected), you can type in a name.
Once you gave a name to a Movie Clip symbol, it become a unique instance; by calling the instance name in AS3.0, we can add all kinds of functions and events to it.

Action Script 3.0 is CASE SENSITIVE!

One thing you need to pay attention to is that do not start any instance name with a capital letter; the main reason is that there are many keywords in AS3.0 starting with capital letter. An instance name starting with capital letter would confuse Flash and cost all types of strange errors.
For the purpose of organization and debugging, especially when you work with a team, if your code is written with random variable names, it will be a big headache when you or your colleagues try to revisit/fix your code.
The major advantages of conforming to a certain set of naming convention are:
1. To reduce the effort needed to understand your code
2. To help formalize expectations and promote consistency within a development team
3. To enhance clarity in cases of potential ambiguity
4. To enhance the aesthetic and professional appearance of work product (wikipedia)

The typical naming convention in AS3.0 is usually like this:
If you have a stage element which will function as an interactive button, you should start with btn/button (usually, you do not start an instance with capital letter) then goes the name you have for your button.For example, a round shape button should be named "btnCircle"; if you have more than one same kind of buttons on stage, you should add a number after the name like "btnCircle1" or "btnCircle2"
If you have a movie clip element, you should start with mc/m. For example, a movie clip contains an animation of jumping bunny we could name it "mcBunny" or "mcJumpBunny"
If you have multiple scenes, then elements on different scenes should be able to be identified from the name. For example, a movie clip contains a ball in scene 1 could be named "mcBall_Scene1"
Last but not least, an instance name cannot contain special characters such as exclamation point, question mark, etc.

Dot Syntax and Accessing Properties



In AS3.0 everything is structured with the Dot Syntax. Basically, it starts with your instance name, next a dot, and then you put properties/functions. So we have two parts in our Dot Syntax, the first part refers to which elements we want to access and the second part refers to what we want to do with it. To put it simple, you can think of the first part as "the who" and the second part as "the what"

*To follow along the workshop, please download **flash_lesson2.zip**, extract to your desktop and open **0_dot_syntax.fla**

For examples, the instance name of the yellow rectangle is "mRectangle" if we want to reduce the opacity to 50% , we can use the alpha property.

```
mRectangle.alpha = .5;
```

Note that every executable line in AS3.0 should be ended with a semi-colon.

More common properties can be find under the action script panel in Flash, AS3.0>>Flash.Display>>MovieClip>>Properties

What is a Function?

A function is a reusable block of code that will perform certain actions.

*To follow along the workshop, please open **1_functions.fla**

To create your own function, simply type in:

```
function myFunction():void {    trace("My function is working!!")    }
```

To use the function you just wrote, just type in

```
myFunction();
```

Note here that **trace()** is a build-in function come with AS3.0 that displays certain values in output window; **stop()** is also a build-in function that uses to stop the playback of the timeline.

Events and Event Listener

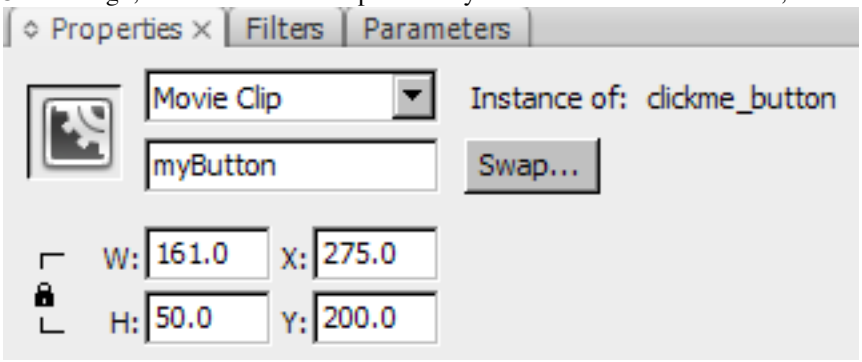
If we ever want to add interactivities to the buttons, we need to define an Event that happens and link the event to our buttons. So every time we interact with the buttons by clicking or hovering, it does the something we expect to happen.

For example, if we want to create button that directs us to a website when we click on it, we need to create an event that direct us to a website, then we create a button, then we add the event to the button.

*To follow along the workshop, please open **2_URLRequest.fla**

Making a Button

On the stage, we have a movie clip with "myButton" as its instance name;



These lines in the action panel make "myButton" act like a button; that is when you hover your mouse on it, the cursor turns into a hand icon.

```
myButton.buttonMode = true;  myButton.mouseChildren = false;
```

URL Request

Next, we write a function named "clickHandler" that directs the page to a certain web address. **"http://www.sfu.ca" navigateToURL(myLink, "_blank")** specifies the URL and as well as the target frame of the action(just like the target frame properties in HTML).

```
function clickHandler(evt:MouseEvent):void  {        var
myLink:URLRequest = new URLRequest ("http://www.sfu.ca");
        navigateToURL(myLink,"_blank");  }
```

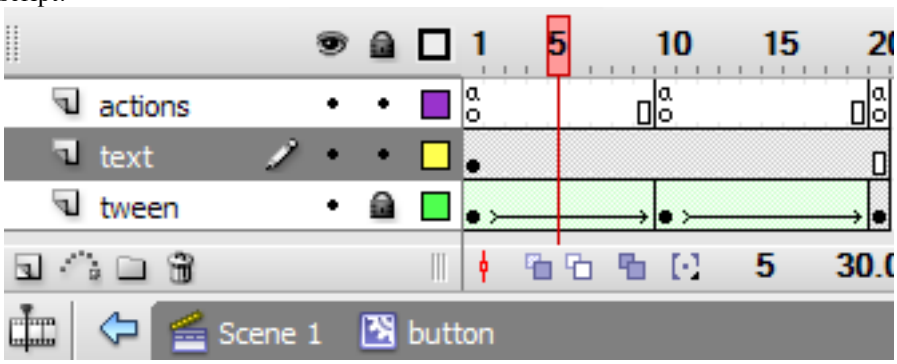
The last step is to add an event listener to our button, so it will perform the function when we click on it.

```
myButton.addEventListener(MouseEvent.CLICK, clickHandler);
```

Basic Timeline Control

*To follow along the workshop, please open **3_Events_Buttons.fla**

You probably have noticed that in each one of my files, all the codes are placed on the first layer and there is nothing on the action layer. The reason I am doing this is that when a SWF file being loaded onto your computer, it begins with the top layer. Since my codes might alter the stage elements later, you do not want you audience to see the pre-composed state of your project. More importantly, every time I visit my code it will save me a huge amount of time on searching if all codes are on the top layer.
Enough talking on project organization, let us get back on the coding. In this file, we want to add roll over/out effects to our button and also make it control the timeline of the bouncing ball.
1. Double click on the movie clip "button" in the library to open the movie clip symbol; basically, it is an animation of the background color goes from grey to green then comes back to grey again. On each keyframe, there is a stop() function to make it stop, so we will be able to control it with our own action script.



2. In the action panel, these codes defines the function for roll over/out effects through controlling animation playback inside of "myButton"

```
function buttonOver(evt:MouseEvent):void{     trace("mouse over");
        myButton.gotoAndPlay(2);  }      function
buttonOut(evt:MouseEvent):void{       trace("mouse out");
        myButton.gotoAndPlay(13);  }
```

3. Next, let us add the roll over/out effects to "myButton" using event listener;

```
  myButton.addEventListener(MouseEvent.ROLL_OVER,buttonOver);
myButton.addEventListener(MouseEvent.ROLL_OUT,buttonOut);
```

4. Then we need to have a function that controls the timeline in the "mcBouncing" movie clip, so when we click on the button the animation will play. Also, we need to put a stop in the bouncing animation.

```
function buttonClick(evt:MouseEvent):void{    trace("click!");
        mcBouncing.gotoAndPlay(2);  }
```

5. Finally, link the timeline control function to "myButton".

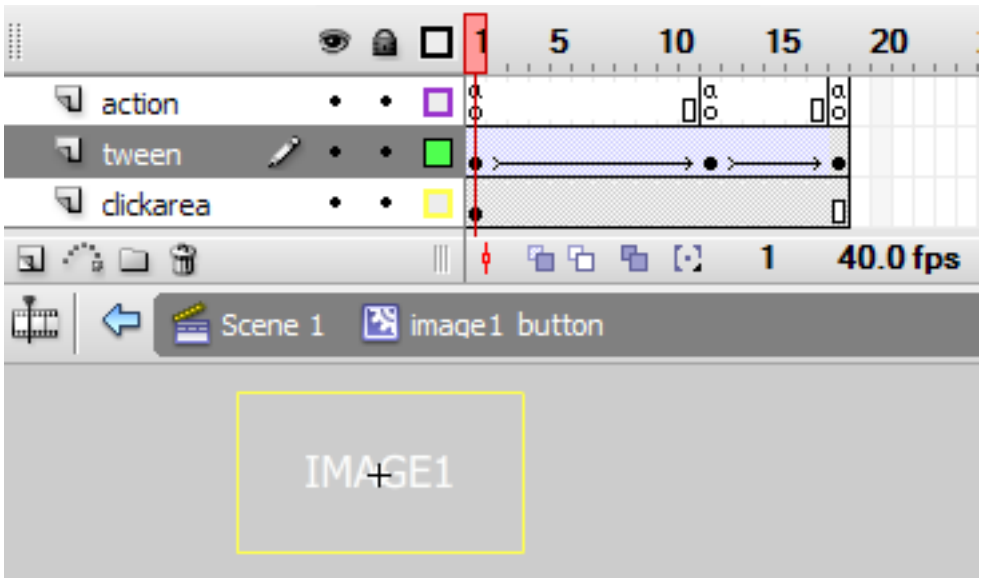
```
myButton.addEventListener(MouseEvent.CLICK,buttonClick);
```

Exercise: Complete the Flash Gallery

In the final part of this workshop, you are going to use the knowledge you have learned on AS3.0 and complete the flash gallery.
*To follow along the workshop, please open **4_BuildingYourOwnFlashSite.fla**
Two fully function buttons has been created, you will need to turn the static texts "IMAGE3", "IMAGE4" and "IMAGE5" into interactive buttons. Also, you need to customize the roll over/out effects of those buttons, then add the image loader function to the buttons and make them load different images.
Before you start it, here is a brief explanation
1. Creating buttons

```
btnImage1.buttonMode = true;  btnImage1.mouseChildren = false;
  btnImage2.buttonMode = true;  btnImage2.mouseChildren = false;  //
Making buttons function like a button
  function btnOver(evt:Event):void {  evt.target.gotoAndPlay(2);  }
  function btnOut(evt:Event):void {   evt.target.gotoAndPlay(13);  }
// Defining the events for rollover/rollout effects
```

Notes that **evt.target** will automatically deprive the instance name and place it in the function
2. Create roll over/out effects using tween animation



3. Write a function to control the ROLL_OVER/ROLL_OUT effects;

```
function btnOver(evt:Event):void
 {
evt.target.gotoAndPlay(2);
  }
    function btnOut(evt:Event):void
 {      evt.target.gotoAndPlay(13);
  }

// Defining the events for rollover/rollout
  btnImage1.addEventListener(MouseEvent.ROLL_OVER, btnOver);
btnImage1.addEventListener(MouseEvent.ROLL_OUT, btnOut);
  btnImage2.addEventListener(MouseEvent.ROLL_OVER, btnOver);
btnImage2.addEventListener(MouseEvent.ROLL_OUT, btnOut);

// Adding the rollover/out events to btnImage1
```

4. Place each image in different FLA files, create a intro-animation (such as fading in effects) and export as SWF files.

5. Create the function to loading external SWF files to stage

```
function imageLoader(evt:MouseEvent):void {
var myrequest:URLRequest = new URLRequest(evt.target.name + ".swf");

 // external image name must be "yourInstanceName + .swf"
        var myloader:Loader = new Loader();
        myloader.load(myrequest);
        stage.addChild(myloader);
        myloader.x = 0;
        myloader.y = 72;
  }
```

6. Add the loading Function to MouseClick Event

```
btnImage1.addEventListener(MouseEvent.CLICK, imageLoader);
btnImage2.addEventListener(MouseEvent.CLICK, imageLoader);
```