

## Introduction to Flash Final Project

Using homework five as your start files create a Flash based website that incorporates images, sound, video, dynamic text, preloader and components. Remember to check your code for any errors. It must be letter perfect. *Note: All ActionScript not explicitly explained in this exercise file is covered in detail in the week 13-15 archives.* [See sample final project here](#). You will find files to download within the project instructions.

1. The following modifications will be made to the homework five files:
  - a. Base movie adds sound controls and a pre-loader
  - b. Main section adds input and dynamic text
  - c. Section one adds dynamic text which uses the load.data command to import a separate text file. It uses ActionScript to control scrolling.
  - d. Section two implements a photo gallery through the use of arrays
  - e. Section three uses the FLVPlayback component to load video files (.flv)
  - f. Many section movies use components.
  
2. Design Assessment Rubric- 5 points (1 point each)
  - a. Typography: Should be consistent and easy to read.
  - b. Images/Sounds: Images should be displayed with appropriate resolution. Sounds should start playing quickly.
  - c. Layout: Should follow a consistent pattern and appear seamless.
  - d. Hyperlinks: ALL should function correctly.
  - e. Navigation: Easy and consistent.
  
3. Technical assessment- 15 points  
All of the technical components need to work similar to the functionality seen in the demonstration project.
  - a. Base movie: Pre-loader, entry/welcome, and sound controls, input and dynamic text fields (6 points) *Note: Main movie changes are minimal.*
  - b. Section One- Dynamic text data loaded from external file (3 points)
  - c. Section Two- Photo Gallery (images and titles) (3 points)
  - d. Section Three- Video Gallery (3 points)Bring in to class on the final day or email all the working files to me. *Subject: Your name Flash A Final Project.* **No submissions will be accepted past the due date.**

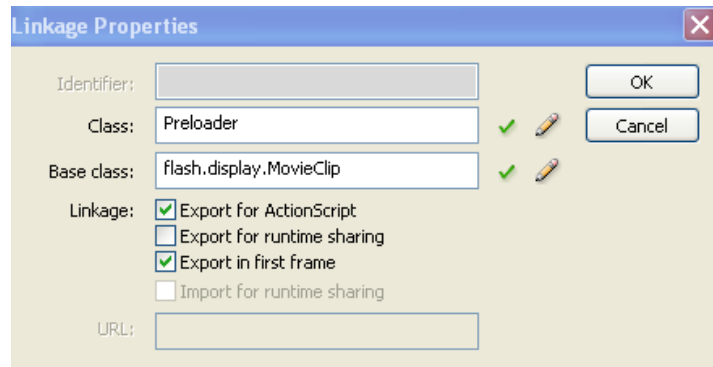
### ***Base Movie- Creating a preloader***

Animated preloaders give the user precise information about the progress of the loading process. This animation needs to communicate graphically what percentage of the site has loaded. The preloader is the first element someone will see when visiting your site. I have created a very simple pre-loader that you can simply drag and drop on to the first frame of your base movie.

1. Create a new folder labeled final\_project with the following sub-folders:
  - a. images
  - b. sounds
  - c. text
  - d. video

2. Open your hw5 base movie and save to your final\_project folder as final\_project.fla
3. [Download assets.fla here](#) and save to your new final\_project folder
4. Select File> Import > Open External Library> assets.fla
5. Select the "Preloader" movie clip in the assets Library and drag it to the stage of your final\_project.fla.

6. Remove "Preloader" from the stage. It is still in the Library.
7. In final\_project.fla Library, right-click on the Preloader movie clip select Linkage. In the Linkage Properties select "Export for ActionScript". This automatically puts a checkmark in the box "Export in first frame". Make sure the word Preloader shows up next to Class. Click OK.



8. Select frame one of the actions layer and replace the existing code with the code written below. Note: When using this preloader, if you choose Test Movie and get an error message when you choose Simulate download, you can ignore this.

```

stop();

var myLoader:Loader = new Loader();
myLoader.contentLoaderInfo.addEventListener(Event.OPEN, showPreloader);
myLoader.contentLoaderInfo.addEventListener(ProgressEvent.PROGRESS, showProgress);
;
myLoader.contentLoaderInfo.addEventListener(Event.COMPLETE, showContent);

var myPreloader:Preloader = new Preloader();

function showPreloader(event:Event):void {
    addChild(myPreloader);
    myPreloader.x = stage.stageWidth/2;
    myPreloader.y = stage.stageHeight/2;
}

function showProgress(event:ProgressEvent):void {
    var percentLoaded:Number = event.bytesLoaded/event.bytesTotal;
    myPreloader.loading_txt.text = "Loading - " + Math.round(percentLoaded * 100)
} + "%";
    myPreloader.bar_mc.width = 198 * percentLoaded;
}

function showContent(event:Event):void {
    gotoAndStop(5);
    removeChild(myPreloader);
    addChild(myLoader);
}

```

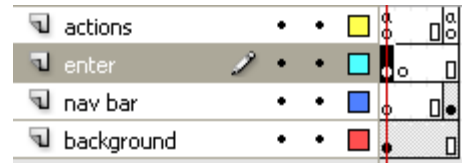
This code uses the contentLoaderInfo property of the Loader class to monitor the progress of the all the content being loaded into your base movie. The line "var myPreloader:Preloader = new Preloader" creates a new class "Preloader" that extends the MovieClip class and gives it all the properties and methods of the MovieClip. This has been accomplished in step 7 above. We then create the "showPreloader" function which adds the preloader object, and positions it on the stage. The "showProgress" function uses

properties of the Loader class to determine how much data has been loaded. The “showContent” function then moves the playback head to frame 5, removes the preloader, and uses addChild to add myLoader to the display list. The actual item to inhabit myLoader will be main.swf but it will not be called to the stage until later but the stage is set.

## Base-Movie Using Input and Dynamic Text

On the main timeline do the following:

1. Add frames in the background layer until frame 5 and a keyframe in frame 5 of the actions layer.
2. Click and drag the keyframe in the nav bar to frame 5.
3. Create a new layer called enter and drag mcWelcome from the **assets.fla** Library to frame one of the enter layer.



4. Inset a blank keyframe in frame two and a frame in frame 5. Your timeline should look like this:
5. Double-click on mcWelcome on the stage and type the code below on the actions layer. This is the same thing you did in chapter 13; exercise 5 for the Welcome message after the user enters their name.

6. Frame One

```
stop();
var visitorName:String;
function enterName(event:MouseEvent):void
{
    if(name_txt.text != "")
    {
        visitorName = name_txt.text;
        nextFrame();
    }
    else
    {
        name_txt.text = "Please enter your name.";
    }
}
enter_btn.addEventListener(MouseEvent.CLICK, enterName);
```

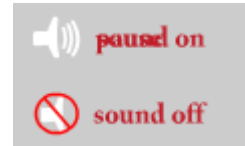
7. Frame Two

```
function enterSite(event:MouseEvent):void {
    parent.myLoader.load(new URLRequest("main.swf"));
    message_txt.text = "";
}
enterSite_btn.addEventListener(MouseEvent.CLICK, enterSite);
message_txt.text = "Welcome, " + visitorName + ". Thank you for visiting our web site!";
```

**Note:** While the embedding is already done with the mcWelcome text fields, all future dynamic text fields you create need to have characters embedded. With the text field selected click on the Embed button in the Property Inspector and select Uppercase and Lowercase.

## Base Movie- Creating sound buttons and Loading Sound

1. Lock the background and enter layers.
2. Select frame five of the nav bar layer.
3. From the assets.fla Library select soundOff, soundOn, soundPause and drag them above the nav\_mc movie clip.
4. Give them instance names stop\_btn, play\_btn, and pause\_btn.
5. Position the soundPause button directly over soundOn so that it looks like the image to the right.
6. Select frame 5 of the actions layer and write the code below. Note: You will substitute the name of the mp3 file (james\_bond.mp3) on line one for the mp3 file that you choose to use. *The code for this section is explained in the week 14 class archive.* Now save and test you base movie then publish it.



```
1  var soundReq:URLRequest = new URLRequest("sounds/james_bond_test.mp3");
2  var sound:Sound = new Sound();
3  var soundControl:SoundChannel = new SoundChannel();
4  var resumeTime:Number = 0;
5  sound.load(soundReq);
6  sound.addEventListener(Event.COMPLETE, onComplete);
7  soundControl.addEventListener(Event.SOUND_COMPLETE, doneSound);
8
9  function doneSound(event:Event) {
10     pause_btn.visible = false;
11     pause_btn.removeEventListener(MouseEvent.CLICK, pauseSound);
12     play_btn.visible = true;
13     play_btn.addEventListener(MouseEvent.CLICK, playSound);
14 }
15
16 function onComplete(event:Event):void {
17     play_btn.addEventListener(MouseEvent.CLICK, playSound);
18     stop_btn.addEventListener(MouseEvent.CLICK, stopSound);
19 }
20
21 function playSound(event:MouseEvent):void {
22     soundControl = sound.play(resumeTime);
23     pause_btn.visible = true;
24     pause_btn.addEventListener(MouseEvent.CLICK, pauseSound);
25     play_btn.visible = false;
26     play_btn.removeEventListener(MouseEvent.CLICK, playSound);
27     soundControl.addEventListener(Event.SOUND_COMPLETE, doneSound);
28 }
29
30 function pauseSound(event:MouseEvent):void {
31     resumeTime = soundControl.position;
32     soundControl.stop();
33     play_btn.visible = true;
34     play_btn.addEventListener(MouseEvent.CLICK, playSound);
35     pause_btn.visible = false;
36     pause_btn.removeEventListener(MouseEvent.CLICK, pauseSound);
37 }
38
39 function stopSound(event:MouseEvent):void {
40     soundControl.stop();
41     play_btn.visible = true;
42     play_btn.addEventListener(MouseEvent.CLICK, playSound);
43     pause_btn.visible = false;
44     pause_btn.removeEventListener(MouseEvent.CLICK, pauseSound);
45     resumeTime = 0;
46 }
47 pause_btn.visible = false;
```

## Base Movie- Enhancing the home button

Unlock the background layer, select both the home\_mc and contact\_mc on the stage and select Edit> Cut

Select keyframe five of the nav bar layer and choose Edit> Paste in place

Double click on the home\_mc

Select frame one of the actions layer and replace the existing code with the code below.

```
function home(event:MouseEvent):void {  
    this.parent.myLoader.load(new URLRequest("main.swf"));  
    this.parent.nav_mc.gotoAndStop(this.parent.nav_mc.start);  
    this.parent.nav_mc.section_one.enabled = true;  
    this.parent.nav_mc.section_two.enabled = true;  
    this.parent.nav_mc.section_three.enabled = true;  
}  
home_btn.addEventListener(MouseEvent.CLICK, home);
```

Originally  
this said  
section one

This code assumes your mcNav symbol on the main timeline instance name is nav\_mc. The enhancement to the home button functionality is to return the movie clip to the starting state, thereby restoring the nav button colors, and restoring the "enabled" state of all the nav buttons.

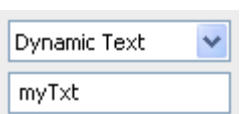
## Main Movie


1. Open main.fla from homework five, remove frames 2 and 3 from all layers **and the content of frame one of the content layer**.
2. Remove all the code from the actions layer.
3. Provide a description of the content the user will see in the sections of your final project on frame one of the content layer then save and publish main.fla

## Section\_one Movie

This movie focuses on using properties to format dynamically loaded text and some serious ActionScript to make the text scroll continuously on press. While it would seem easy to do this with the UIScrollBar component, this component only works in Test Movie mode and not within web browsers. After considerable research I found this excellent solution.

1. Open section\_one.fla from homework five, remove frames 2 and 3 from all layers **and the content of frame one of the content layer**.
2. Remove all the code from the actions layer.
3. In a text editor, create a text file that answers the following questions with at least two sentences per answer, and save the file as actionscript.txt into your text sub-folder.
  - a. What is a variable?
  - b. What is the purpose of a function in ActionScript?
  - c. What is a property of an object and give one example.
  - d. What is an example of a built-in Flash method?
  - e. Give an example of how event listeners work in Flash.
4. Based on chapter 13 exercise 4 use at least three HTML tags to provide formatting to the actionscript.txt file.



5. Select the content layer and create a dynamic text field named myTxt. Set the text field to Multiline and embed Uppercase and Lowercase characters.
6. Select Window> Common Libraries> Buttons> and drag two instances of the "circle button - next" from the Circle Buttons sub-folder of the classic buttons folder within this Library to the stage.
7. Use the transform tool to rotate one button facing up and one button facing down. See picture to the right-----> 
8. In the Properties panel give an instance name scrollUp to the button pointing up and scrollDown to the button pointing down.
9. Select frame one of the actions layer and replace the code with what is written below. *Note you can cut and paste this section of code but you must replace all of the open and close quotes with new ones inside of the ActionScript panel.*

```
//-----loading external text and display it-----
var loader:URLLoader = new URLLoader();
var myRequest:URLRequest = new URLRequest("text/actionscript.txt");
loader.load(myRequest);
loader.addEventListener(Event.COMPLETE, textBox);
function textBox(event:Event):void {
    myTxt.htmlText = event.target.data;
    myTxt.background = true;
    myTxt.backgroundColor = 0xCCCCCC;
    myTxt.border = true;
    myTxt.borderColor = 0xC01D29;
    myTxt.wordWrap = true;
}
//-----make buttons to scroll the loaded text-----
var scrolling:Boolean = false;
var scrollDirection:String;

scrollDown.addEventListener(MouseEvent.CLICK,scrollTextDown);
scrollUp.addEventListener(MouseEvent.CLICK,scrollTextUp);
this.addEventListener(MouseEvent.CLICK,stopScroll);

function scrollTextDown(event:MouseEvent):void {
    scrolling = true;
    scrollDirection = "down";
    myTxt.scrollV +=1;
    this.addEventListener(Event.ENTER_FRAME,checkButtons);
}
function scrollTextUp(event:MouseEvent):void {
    scrolling = true;
    scrollDirection = "up";
    myTxt.scrollV -=1;
    this.addEventListener(Event.ENTER_FRAME,checkButtons);
}
function stopScroll(event:MouseEvent):void {
```

```

        this.removeEventListener(Event.ENTER_FRAME,checkButtons);
        scrolling = false;
    }
    //-----checking to see if buttons are pressed-----
    function checkButtons(event:Event):void {
        if (scrolling) {
            if (scrollDirection == "down") {
                myTxt.scrollIV +=1;
            } else if (scrollDirection == "up") {
                myTxt.scrollIV -=1;
            }
        }
    }
}

```

10. Now save and publish section\_one fla

The code for scrolling the text is broken down into three parts; loading and displaying the external text, making the buttons scroll the text that is loaded, and checking if our buttons are held down.

In the first section we create a URLRequest object and pass the path to our txt file. This object knows where our file is. Next we create an URLLoader object that loads our txt file. We add an Event.COMPLETE to our textLoader, which will fire when our external file is completely loaded. When this happens, we assign the loaded text to the myTxt dynamic text field. This process is handled by the textBox function.

In the "make buttons scroll text" section we first declare two variables, one for keeping track if the text is scrolling (scrolling variable, this is of type Boolean, true or false) and the other for checking in which direction the text is scrolling (up or down). Then we add a MOUSE\_DOWN event to our buttons and a MOUSE\_UP event to the stage.

The scrollTextDown function handles (tell us what should happen when the left button of our mouse is down) the MOUSE\_DOWN event added to scrollDown button. Inside this function we set our scrolling variable to true (because we are scrolling the text), set the scrollDirection variable to "down" (we are scrolling down) and increase the scrollIV property of our text\_field with 1. We also add an ENTER\_FRAME event to the stage.

The scrollTextUp function makes the same thing like scrollTextDown function but for the scrollUp button.

The stopScroll function handles the MOUSE\_UP event added to the stage. When this event happens we remove the ENTER\_FRAME event and setting the scrolling variable to false.

In the third section, "checking if our buttons are pressed and hold down" we have the function that handles the ENTER\_FRAME(fires every frame of our application) event added to the stage. Inside the checkButtons function we use a nested (one inside other) if conditional statement. First we want to know if our text is scrolling (if (scrolling), this is identical with if (scrolling ==true)) and if that is true we want to know in which direction. If is scrolling down

we add 1 to the scrollV property of the text\_field and if is scrolling up we subtract 1 from scrollV property.

## Section\_two Movie

This movie uses the **UILoader** component. In addition, you will add images and text dynamically through the creation of arrays.

1. Open section\_two.fla from homework five, remove frames 2 and 3 from all layers **and the content of frame one of the content layer..**
2. Remove all the code from the actions layer.
3. From the assets.fla file drag six copies of thumb\_btn on to the stage
4. Give them instance names btn1 through btn6
5. Select Window> Components and drag a copy of the UILoader onto the stage
6. Name it my\_ldr
7. Select the text tool
8. Create a dynamic text field set to multi-line, embed characters and name it myTxt
9. Find/create a set of 72 ppi resolution images no bigger than 400 x 400 pixels
10. Place them in the images sub-folder of your final\_project folder

Select frame one of the actions layer and write the following, making sure to substitute your image names and explanations in "myArray" and "desc". Then save and publish section\_two.fla

```
var myArray:Array = ["images/DSC_2031.jpg", "images/DSC_2032.jpg",  
"images/DSC_2033.jpg", "images/DSC_2034.jpg", "images/DSC_2035.jpg",  
"images/DSC_2036.jpg" ]  
  
var desc:Array = ["Charlie Brown Farmer", "Charlie Brown Ocean", "Charlie Brown  
Patissiere", "Charlie Brown Beans", "Charlie Brown Golfer", "Charlie Brown Nature" ]  
  
btn1.addEventListener(MouseEvent.CLICK, ldr1)  
function ldr1(event:Event){  
    my_ldr.source = myArray[0];  
    myTxt.text = desc[0];  
}  
btn2.addEventListener(MouseEvent.CLICK, ldr2)  
function ldr2(event:Event){  
    my_ldr.source = myArray[1];  
    myTxt.text = desc[1];  
}  
btn3.addEventListener(MouseEvent.CLICK, ldr3)  
function ldr3(event:Event){  
    my_ldr.source = myArray[2];  
    myTxt.text = desc[2];  
}  
btn4.addEventListener(MouseEvent.CLICK, ldr4)  
function ldr4(event:Event){  
    my_ldr.source = myArray[3];  
    myTxt.text = desc[3];  
}  
btn5.addEventListener(MouseEvent.CLICK, ldr5)  
function ldr5(event:Event){  
    my_ldr.source = myArray[4];  
    myTxt.text = desc[4];  
}  
btn6.addEventListener(MouseEvent.CLICK, ldr6)  
function ldr6(event:Event){  
    my_ldr.source = myArray[5];  
    myTxt.text = desc[5];  
}
```

## Section\_three Movie

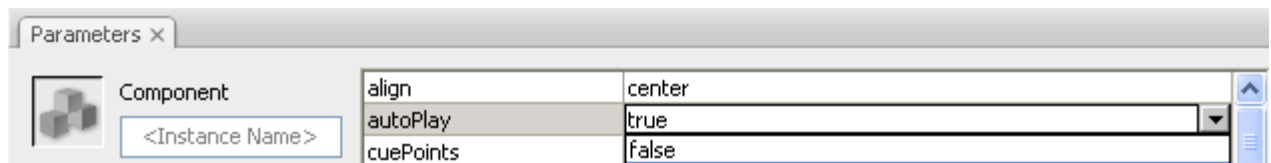
This movie focuses on loading external video files and downloading the video progressively (so the viewer can watch it as it is downloading). We utilize the FLV playback component which lets you control which video plays, whether the video plays automatically, and other aspects of playback. [This section assumes you use the flv files located in this folder here and place them in the videos sub-folder of your final project.](#) *Note: If you choose to use your own videos and convert them to flv files you will earn **8 extra points** as this is an additional learning process that you would be taking on your own.*

You'll turn autoplay off for the initial penguin movie, so that it doesn't begin playing until the viewer presses the Play button. Then you'll change the FLV playback component options at different frames to play the appropriate movies for each button, and turn autoplay on, so that the movies start playing immediately when the button is pressed.

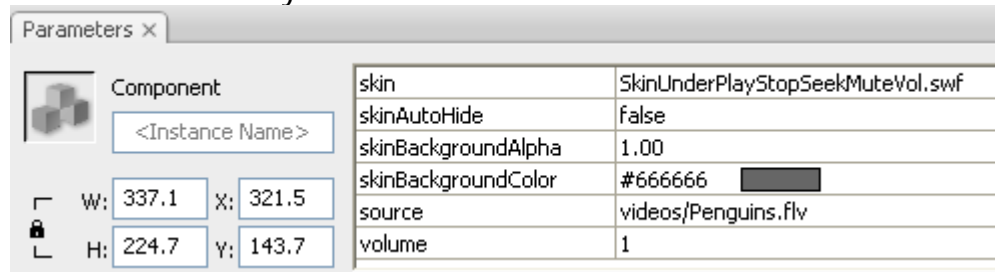
1. Open section\_three.fla from homework five and remove frames 2 and 3 from all layers.
2. Remove all the code from the actions layer.
3. Add the following layers: labels, video player, video buttons.
4. Select frame 20 of all the layers and add frames.
5. Select the labels layer and add keyframes to frames 3, 10, and 16.
6. With the labels layer selected go to the Property Inspector add the following frame labels: penguins, mandrill, tiger to frames 3, 10, and 16 respectively.

## Building the Video Player Layer

1. Select Window > Components and drag the FLVPlayback component from the Video group on to frame one of the Video Player layer. Size it to your liking.
2. Select the Parameters tab.
  - a. Click autoPlay and use the arrow to change the setting from true to false. *Note you will use the arrow to choose your skin and source below.*

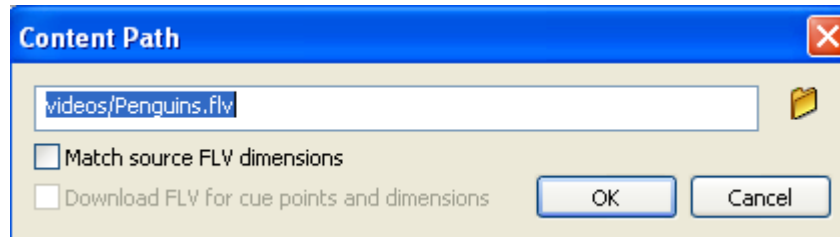


- b. Choose a skin that you like.



- c. Set the source to videos/Penguins.flv and make sure to deselect Match source FLV dimensions for this movie as well as the others that you choose later on. These

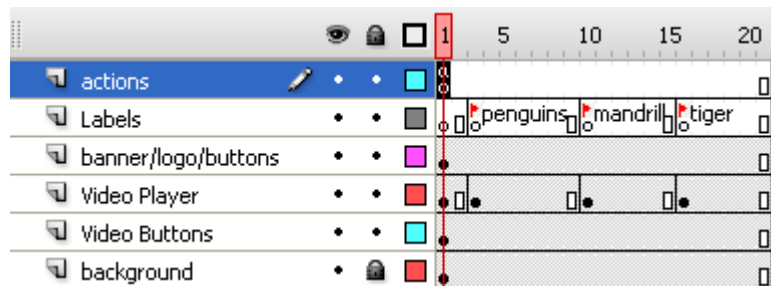
instructions assume that flv files with the proper names are located in your videos folder.



3. Select the Video Player layer and add keyframes on frames 3, 10, and 16.
4. Select the FLVPlayer instance on frame 10, and change the source to videos/Mandrill.flv
5. Select the FLVPlayer instance on frame 16, and change the source to videos/Tiger.flv

### ***Building the Video Buttons Layer***

1. Select File > Import > Open External Library > Assets.fla
2. Drag video\_button1, video\_button2, and video\_button3 on to frame one of the Video Buttons layer
3. With all three selected use the Alignment tab to align their bottom edges
4. Select the buttons and give them instance names video\_button1, video\_button2, and video\_button3 respectively.
5. Your timeline is complete and should look like this:



### ***Scripting Video Buttons***

As you can see, a different video file is associated with each section of the Timeline (Penguins.flv with frame 10, Mandrill.flv with frame 20, and so on). Now you need to write the ActionScript that moves the playhead to the appropriate frame in the Timeline when the button is clicked.

You can use the same set of code for each button, with minor changes. The first line of code adds an event listener, which listens for the button click, and then runs the function (clickListener1, clickListener2, and so on). The function moves the playhead to the appropriate frame in the Timeline. When the playhead moves, the appropriate video file plays, according to the controls in the FLV playback component.

1. Select frame 1 in the Actions layer, and open the Actions panel.
2. Add an event listener for the first button by typing the following line in the Actions panel as below. Then copy and paste the lines you just added two times, and then modify each set for the appropriate video.

3. Your code should look like this:

```
stop();
this.video_button1.addEventListener(MouseEvent.CLICK,clickListener1);
function clickListener1(event:MouseEvent):void {
    gotoAndStop("penguins");
}
this.video_button2.addEventListener(MouseEvent.CLICK,clickListener2);
function clickListener2(event:MouseEvent):void {
    gotoAndStop("mandrill");
}
this.video_button3.addEventListener(MouseEvent.CLICK,clickListener3);
function clickListener3(event:MouseEvent):void {
    gotoAndStop("tiger");
}
```

Save and publish section\_three fla

## ***Testing your Final Project***

Congratulations. You are now ready to test your Final Project. Make sure that you have both saved and published all your movies. Open up your base movie and select Control> Test Movie. Everything should be working. If you have error messages bring them in to the lab or come to my online office hours for assistance. I hope this has been a valuable learning experience for you on your way to become a serious *Flasher*.