Create Button States

One of the big differences between developing applications for a mobile device compared to a desktop computer is handling interaction. On a desktop, your application would make use of the mouse as a primary interaction input, whereas on a device, your fingers do most of the work.

The first thing you will probably realize is that you will not receive any `MouseEvent.ROLL_OVER` events because you cannot actually roll over any objects. Because of this, any buttons that may have rollover states for the desktop computer will not be shown. It is important, and good practice, to make sure that all your buttons have highlighted states, or mouse down states. This will give the users visual feedback that they actually touched on the button that they intended to touch. Creating buttons without proper states may lead the users to think that your application is broken or that the device is frozen.

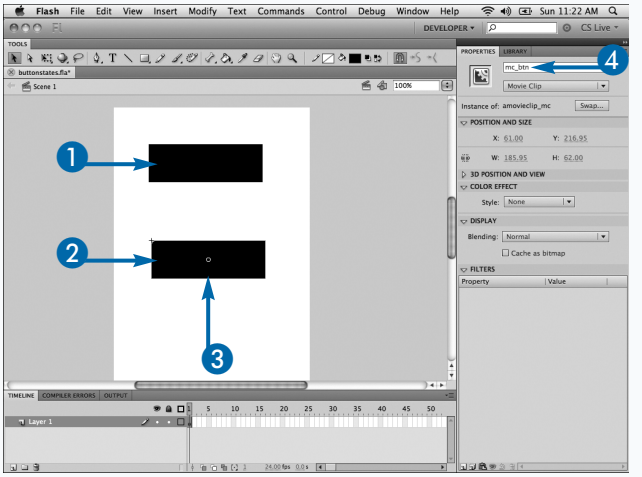
There are two ways to make sure that your buttons have the proper states. The easiest way is to create a `Button` symbol in the Flash IDE and create a new state for your button in the appropriate frame on the Timeline. The second way is to listen for the `MouseEvent.MOUSE_DOWN` event on a `MovieClip` through code. This allows you to change the appearance of your `MovieClip` through code when a finger is touched down on the hit area of your button. If you use this method, you need to ensure that you change the state back when the user lifts his or her finger off the button as well.

Create Button States

1. Create a `Button` symbol.
2. Create a `MovieClip`.

Note: *See Chapter 2, "Getting Started with Flash CS5," for more details on these steps.*

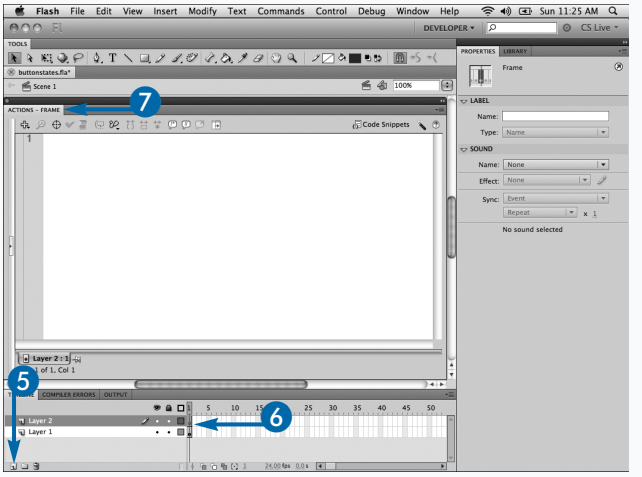
3. Select the `MovieClip` on the Stage.
4. Give the button an instance name, such as `mc_btn`.



5. In the Timeline panel, click the New Layer button.

   A new layer is created.

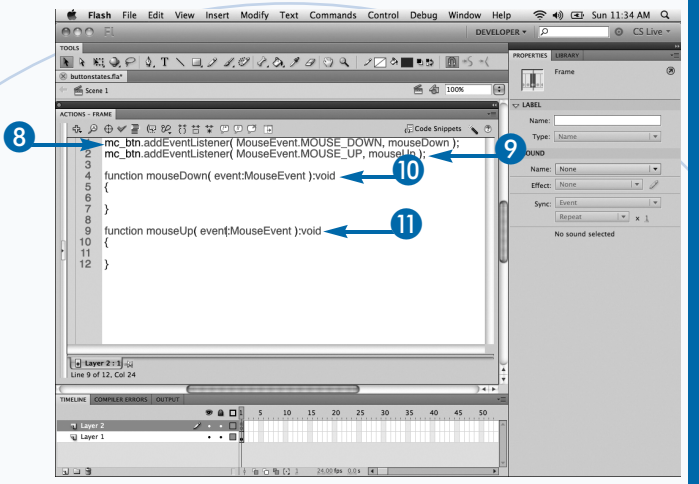
6. Select the new layer.
7. Open the Actions panel.



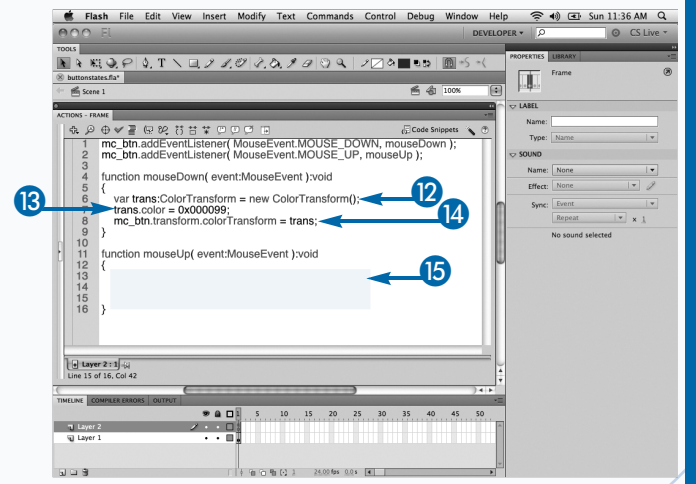
07_904329-ch05.indd 76 8/26/10 10:27 PM

8. Add a mouse down listener to your `MovieClip`, such as `mc_btn.addEventListener( MouseEvent.MOUSE_DOWN, mouseDown );`.
9. Add a mouse up listener to your `MovieClip`, such as `mc_btn.addEventListener( MouseEvent.MOUSE_UP, mouseUp );`.
10. Create a `mouseDown` event handler.
11. Create a `mouseUp` event handler.

Note: *For more details on creating event handlers, see Chapter 2.*



12. In your `mouseDown` function, create a new `ColorTransform` instance, such as `var trans:ColorTransform = new ColorTransform();`.
13. Set the `color` property, such as `trans.color = 0x000099;`.
14. Apply the `ColorTransform`, such as `mc_btn.transform.colorTransform = trans;`.
15. Repeat steps **12** to **14** in your `mouseUp` method and change the color, such as `trans.color = 0x00000;`.
16. Publish the file and click the buttons to see their states change.



Extra

Another thing to keep in mind when creating buttons is size. On a desktop computer, you could have a 1 x 1 pixel button and still be able to click it with the mouse. On a touch screen, your finger is less precise, which makes it really hard to click small buttons. So it is important to make your buttons big enough in size that the users can register a proper touch. It is also a good idea to keep your buttons far enough apart so that the users do not select a button that they did not intend to. When designing buttons in your application, try to give them the same look and feel as the ones that the Android OS uses in its applications. Creating similar buttons will make your users familiar with your application right away. In this section, the example shows creating simple color changes for the different states; however, you can also create a state that shows a circular highlight representing where the finger pressed on the screen.

07_904329-ch05.indd 77 8/26/10 10:27 PM

Respond to Touch Events

For developing for platforms that can handle detecting a user's touch, Adobe has introduced the `flash.events.TouchEvent` class. This class is new and available on Flash Player 10.1, AIR 2.0, and the Android. Touch events can be thought of as mouse events for touch-enabled devices, in that they allow you to respond to basic touch interactions.

When a user interacts with the screen with a single or multiple fingers, multiple types of `TouchEvent`s will be fired that allow you to respond correctly to the user. You can listen for `TouchEvent`s on any `InteractiveObject`. Here is a typical sequence of events that will be fired during a single touch interaction: First, a `TouchEvent.TOUCH_BEGIN` event is fired when the user first presses his finger on the screen of the device. Second, a `TouchEvent.TOUCH_MOVE` is fired if the user drags his finger on the screen while the object is still pressed. Lastly, a `TouchEvent.TOUCH_END` event is fired when the user lifts his finger off the screen.

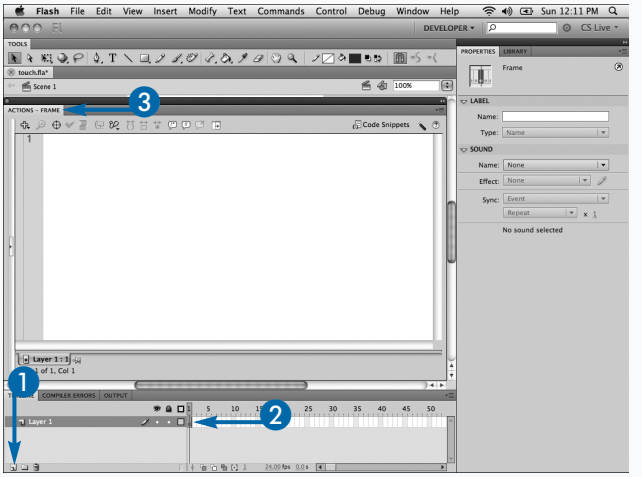
In order to respond to these events, you need to add a listener to an `InteractiveObject`. This can be a `Button`, a `MovieClip`, or even the root Stage of your application. In the example below, I add a few listeners to the main Timeline of a blank `.fla` to create a very simple paint program to help illustrate what I have talked about.

If you are going to develop your application to support platforms other than the Android, you can detect to see if `TouchEvent`s are supported on the device. `flash.ui.Multitouch.supportsTouchEvents` returns true if the device does support `TouchEvent`s and false if it does not. Planning ahead of time to support multiple platforms is never a bad thing; it will save you development time down the road.

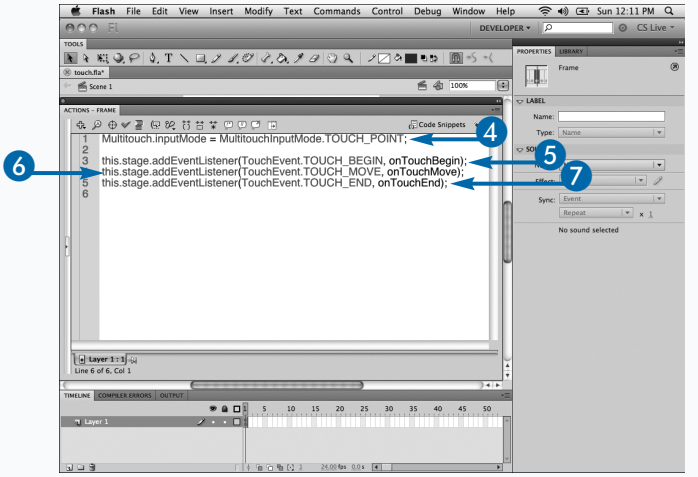
Respond to Touch Events

1. In the Timeline panel, click the New Layer button.

   A new layer is created.
2. Select the new layer.
3. Open the Actions panel.

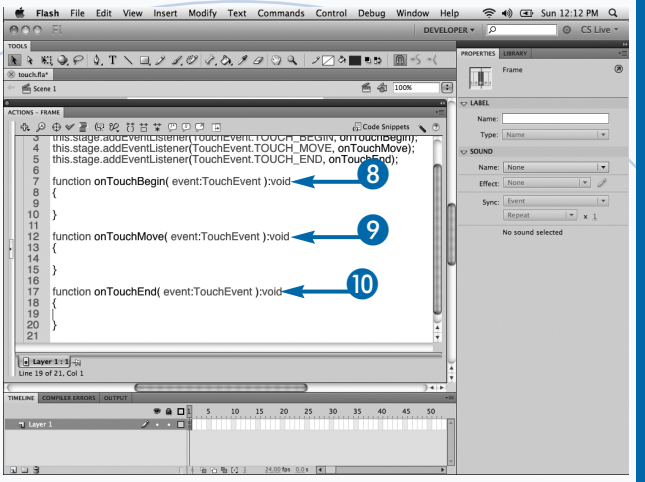


4. Set the multitouch input mode to touch, such as `Multitouch.inputMode = MultitouchInputMode.TOUCH_POINT;`.
5. Add a `TOUCH_BEGIN` listener to the Stage, such as `this.stage.addEventListener( TouchEvent.TOUCH_BEGIN, onTouchBegin );`.
6. Add a `TOUCH_MOVE` listener to the Stage, such as `this.stage.addEventListener( TouchEvent.TOUCH_MOVE, onTouchMove );`.
7. Add a `TOUCH_END` listener to the Stage, such as `this.stage.addEventListener( TouchEvent.TOUCH_END, onTouchEnd );`.

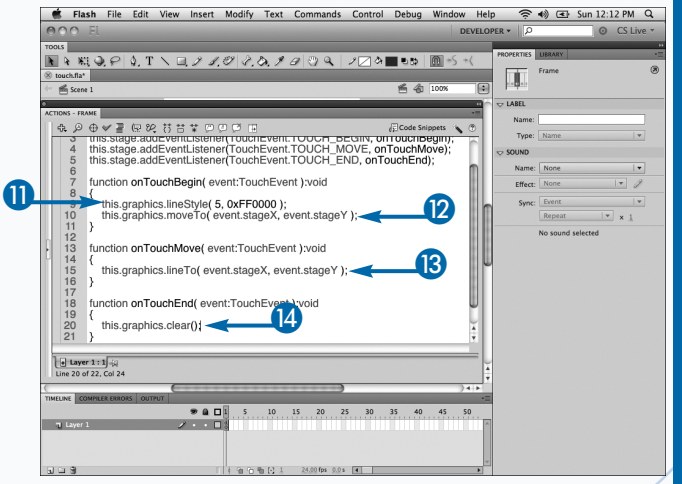


07_904329-ch05.indd 78 8/26/10 10:27 PM

8. Create an `onTouchBegin` event handler for your listener.
9. Create an `onTouchMove` event handler for your listener.
10. Create an `onTouchEnd` event handler for your listener.



11. In the `onTouchBegin` method, set the line style, such as `this.graphics.lineStyle( 5, 0xFF0000 );`.
12. Move the drawing position to the touch location, such as `this.graphics.moveTo( event.stageX, event.stageY );`.
13. In the `onTouchMove` method, draw a line to the new touch location, such as `this.graphics.lineTo( event.stageX, event.stageY );`.
14. In the `onTouchEnd` method, clear the drawing, such as `this.graphics.clear();`.
15. Publish the file and install the application on your device.



Extra

It is really important to plan ahead when creating your applications. Even if at the time of development, you do not plan on supporting any other platforms, you never know when Adobe will support a new platform that you did not expect during development. The less amount of time it takes to get your application working on a new platform, the better. If you do not plan to support multiple touches or gestures in your application, you may want to consider using `MouseEvent`s instead of `TouchEvent`s. This will allow your application to be compatible with more platforms than touch-enabled ones.

The sequence of events is similar to those used in the example in this section. `MouseEvent.MOUSE_DOWN` is fired when the user presses down on the mouse button, `MouseEvent.MOUSE_MOVE` is fired when the user moves the mouse with the mouse button still pressed, and `MouseEvent.MOUSE_UP` is fired when the user releases the mouse button. If you have previous experience developing Flash applications, this concept is not new to you, and you can apply that knowledge the same way you would when developing applications for the Web or desktop.

07_904329-ch05.indd 79 8/26/10 10:27 PM

Track Multiple Touches

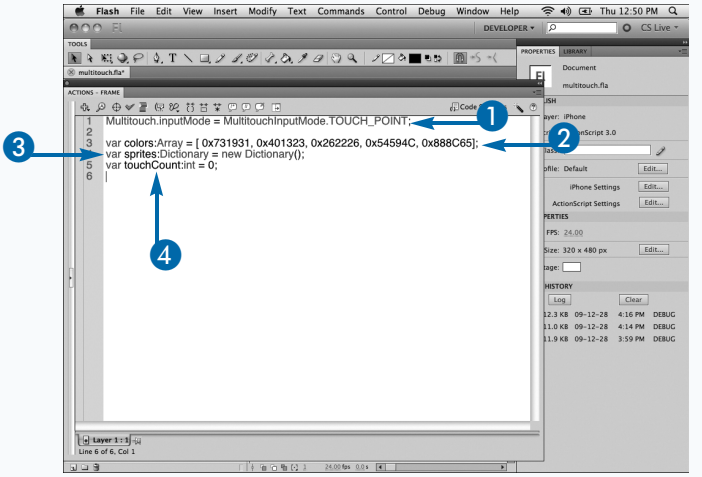
Tracking multiple touches is not much harder than tracking a single touch. The `TouchEvent.touchPointID` property is a unique ID for each unique touch that occurs. This ID is assigned when the `TouchEvent.TOUCH_BEGIN` event is fired and can be used to track unique touches on the screen.

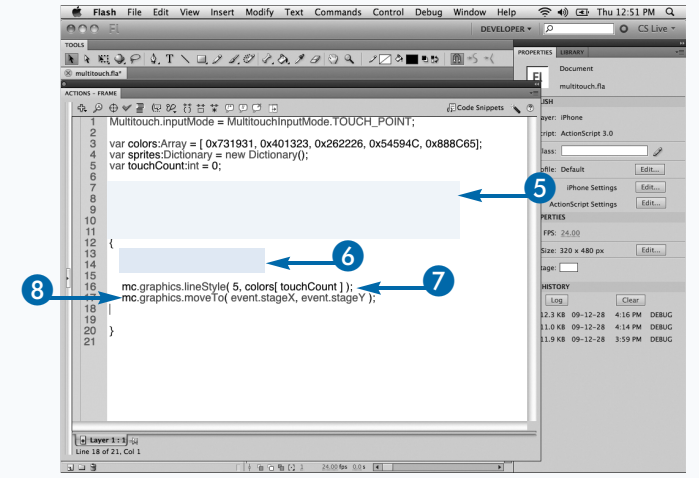
For each new touch that is detected, the `touchPointID` is incremented. For example, if you were to place two fingers on the screen and drag them around, you would receive touch event objects with `touchPointID` values of 1 and 2. If you lifted those fingers and placed them back on the screen, you would get `touchPointID` values 3 and 4. This makes it a little harder to track specific fingers because `touchPointID` 2 does not necessarily mean finger 2. In order to track specific fingers, you will need to store which `touchPointID`s are currently being used in an `Array` or `Dictionary` object, as you will see in the example below. You place the ID in this object during the `TouchEvent.TOUCH_BEGIN` event and remove it in the `TouchEvent.TOUCH_END` event.

Each touch screen device has a different number of touches that it can detect at once. Each Android device may support a different maximum touch. Currently, the Google Nexus One phone supports only 2. If you are planning on releasing your application on multiple platforms that support touch screens, you will want to plan for this number to change and be able to accommodate fewer or more touches. To do this, you can use the `Multitouch.maxTouchPoints` property to determine how many touches can be detected at the same time.

Track Multiple Touches

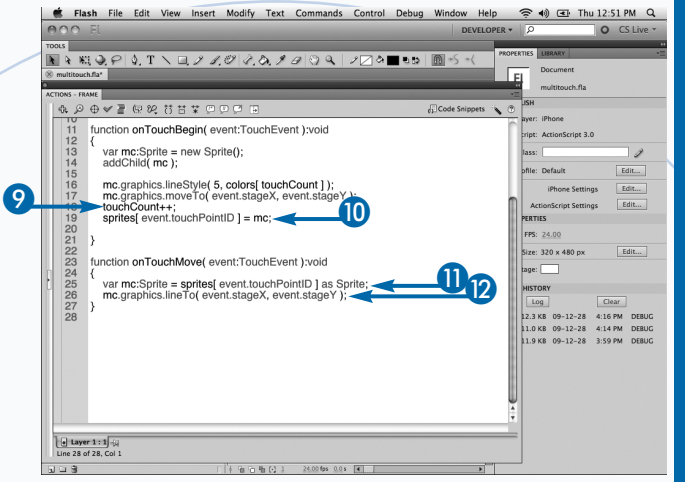
1. Set the input mode, such as `Multitouch.inputMode = MultitouchInputMode.TOUCH_POINT;`.
2. Create an array to hold a color for each finger, such as `var colors:Array = [ 0x731931, 0x401323, 0x262226, 0x54594C, 0x888C65];`.
3. Create a `Dictionary` instance to hold sprite references, such as `var sprites:Dictionary = new Dictionary();`.
4. Create a touch counter, such as `var touchCount:int = 0;`.
5. Add listeners and event handlers for the `TOUCH_BEGIN`, `TOUCH_MOVE`, and `TOUCH_END` events.
6. Create a new `Sprite` instance and add it to the Stage, such as `var mc:Sprite = new Sprite(); addChild( mc );`.
7. Set the sprite's `lineStyle`, such as `mc.graphics.lineStyle( 5, colors[ touchCount ] );`.
8. Set the initial drawing location to the touch position, such as `mc.graphics.moveTo( event.stageX, event.stageY );`.



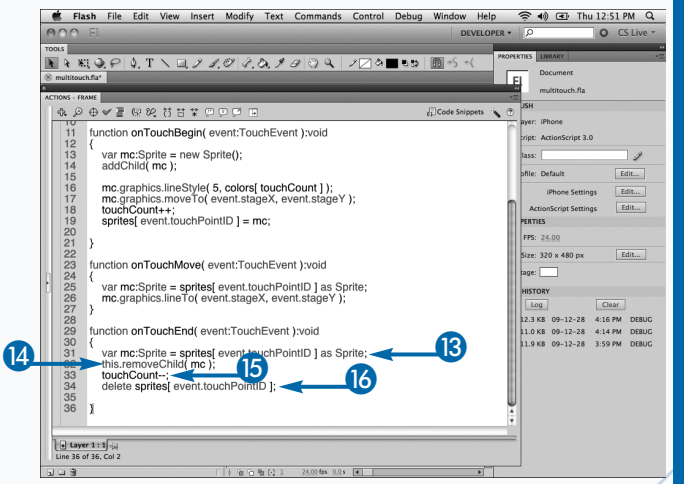


07_904329-ch05.indd 80 8/26/10 10:27 PM

9. Increment the touch count, such as `touchCount++;`.
10. Store a reference to the sprite, such as `sprites[ event.touchPointID ] = mc;`.
11. Get the reference to the sprite, such as `var mc:Sprite = sprites[ event.touchPointID ] as Sprite;`.
12. Draw a line to the new touch position, such as `mc.graphics.lineTo( event.stageX, event.stageY );`.

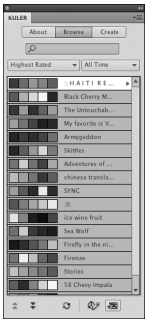


13. Get the reference to the sprite, such as `var mc:Sprite = sprites[ event.touchPointID ] as Sprite;`.
14. Remove the sprite from the Stage, such as `this.removeChild( mc );`.
15. Decrement the touch count, such as `touchCount--;`.
16. Delete the reference to your sprite, such as `delete sprites[ event.touchPointID ];`.
17. Publish the file and install it on your device. Draw with multiple fingers on the screen.



Extra

If you would like to play around with a different color scheme, you can easily find new ones in the Kuler panel. You can access this panel by clicking Window ➔ Extensions> ➔ Kuler in Flash Professional CS5. On this panel, you can browse different color schemes that other users have created. When you have found a scheme that you like, you can add the colors to the Swatches panel. If you do not find one that suits you, you can always edit an existing one or create a new one.



07_904329-ch05.indd 81 8/26/10 10:27 PM

Respond to Zoom Events

One of the main user interactions for multitouch devices is the pinch and zoom gesture. It is most commonly used to scale objects up and down. You have probably used this gesture yourself a number of times to zoom in and out of Web sites in the browser or email on your device. If you are new to the platform, the zoom gesture, sometimes referred to just as *pinch,* is achieved by placing two fingers on the screen and moving them apart to zoom in and moving them closer to zoom out. It would be possible to achieve this effect by tracking multiple touch points, but that is harder than it sounds. Luckily, Adobe has provided the `TransformGestureEvent.GESTURE_ZOOM` event, which will take care of all the hard work.

In order to respond to this gesture, you simply add an event listener to any `InteractiveObject`, just as you have done in previous examples. When the event is fired, the event handler method will be called with a `TransformGestureEvent` object as a single argument.

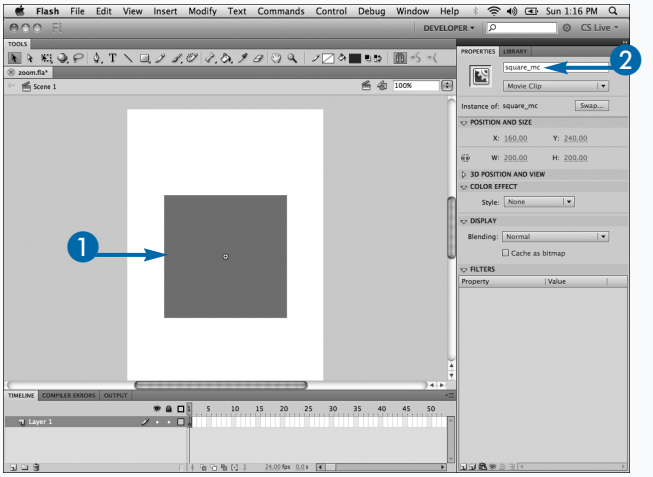
This event contains all the information you need in order to scale the object in relation to a gesture. In particular, the two properties of interest are `scaleX` and `scaleY`. Your first thought may be to simply set the scale values that are returned from the event to your object, but that will give you an undesired result. The reason for this is that the scale properties that are returned are values based on the previous gesture event — not its current scale value. In order to calculate your object's new scale properties, you simply multiply the object's current scale by the event's scale values.

Respond to Zoom Events

1. Create a `MovieClip`.

Note: *See Chapter 2 for more details on this step.*

2. Give it an instance name, such as `square_mc`.

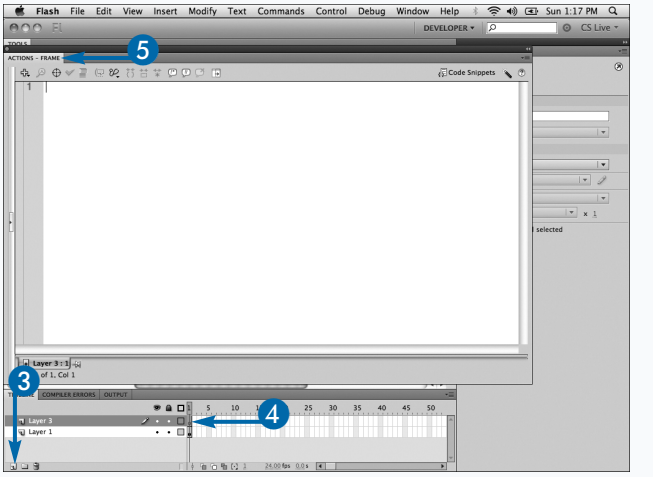


3. In the Timeline panel, click the New Layer button.

   A new layer is created.

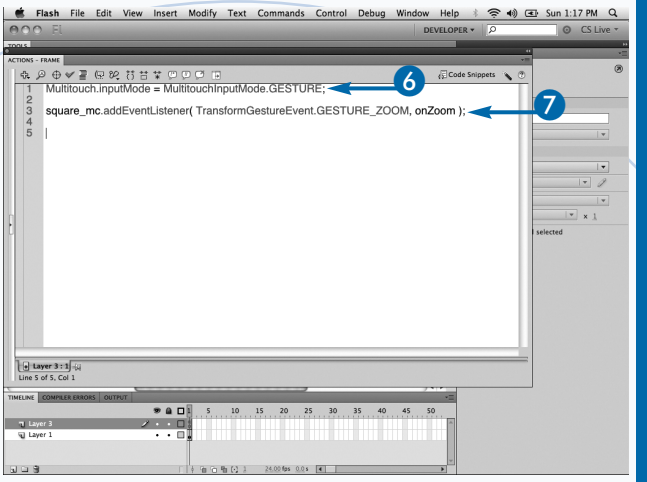
4. Select the new layer.

5. Open the Actions panel.



07_904329-ch05.indd 82 8/26/10 10:27 PM

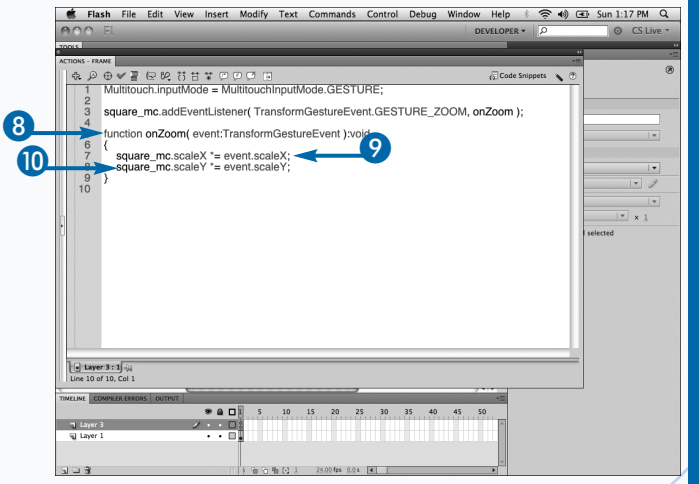
6. Set the input mode, such as `Multitouch.inputMode = MultitouchInputMode.GESTURE;`.
7. Add a listener for the zoom event, such as `square_mc.addEventListener( TransformGestureEvent.GESTURE_ZOOM, onZoom );`.



8. Create an event handler for your zoom listener.

Note: *See Chapter 2 for more details on creating event handlers.*

9. Set the `scaleX` of the square based on the gesture, such as `square_mc.scaleX *= event.scaleX;`.
10. Set the `scaleY` of the square based on the gesture, such as `square_mc.scaleY *= event.scaleY;`.
11. Publish the file and install it on your device.
12. Place two fingers close together on the square and move them apart.



Extra

A tip for creating smooth-looking zoom gestures is to make sure that your registration point is at the center of your object instead of the default top left, or 0,0. This will ensure that your object is scaled from its center and will be consistent with all other objects that react to this gesture in other applications on your device. There are a couple of different ways that you can set the registration point. The easiest is way is to set it in the Convert to Symbol dialog box by selecting the center dot as the registration point. The other way is through code and will require you to have your object in a parent `DisplayObjectContainer`:

```
child.x = -child.width;
child.y = -child.height;
```

This will center the child object to its parent's registration point, assuming that it is at 0,0.

07_904329-ch05.indd 83 8/26/10 10:27 PM

Respond to Rotate Events

Another commonly used multitouch interaction, aside from the pinch and zoom gesture, is the two-finger rotate gesture. To rotate an object, place one finger on it and rotate a second finger around the first. It is important to note that only one finger needs to be on the object that you are rotating. The second finger can be outside of the hit area of the object, but it must remain on the screen at all times.

In order to detect for this gesture, you can listen for the `TransformGestureEvent.GESTURE_ROTATE` event on any `InteractiveObject` on the Stage. If a rotate gesture is detected, this event will be fired, and your event handler will be called with a `TransformGestureEvent` object as its argument. This object contains all the information that you need in order to rotate your object. The `TransfromGestureEvent.rotation` property contains the rotation of the object since the previous rotation event. In order to have this affect your object, you add the event's rotation value to the current rotation value of your object.

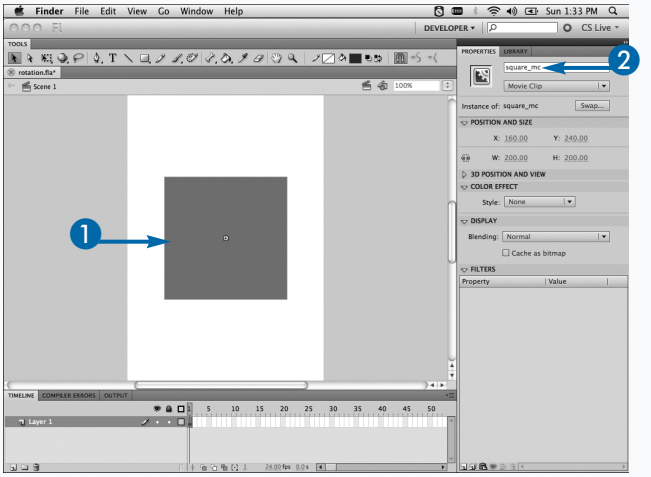
It is a good idea for all the objects that respond to a rotation gesture to have their registration be at their center. This will ensure that your fingers are always on the object as you rotate it. If you had your object's registration point at the default top left, or 0, 0, you could rotate the object off the screen, which could produce some undesirable results.

Respond to Rotate Events

1. Create a `MovieClip`.

Note: *See Chapter 2 for more details on this step.*

2. Give it an instance name, such as `square_mc`.

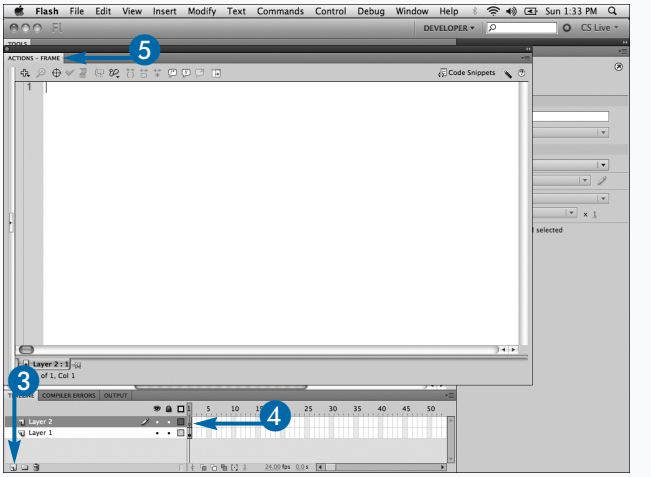


3. In the Timeline panel, click the New Layer button.

   A new layer is created.

4. Select the new layer.

5. Open the Actions panel.

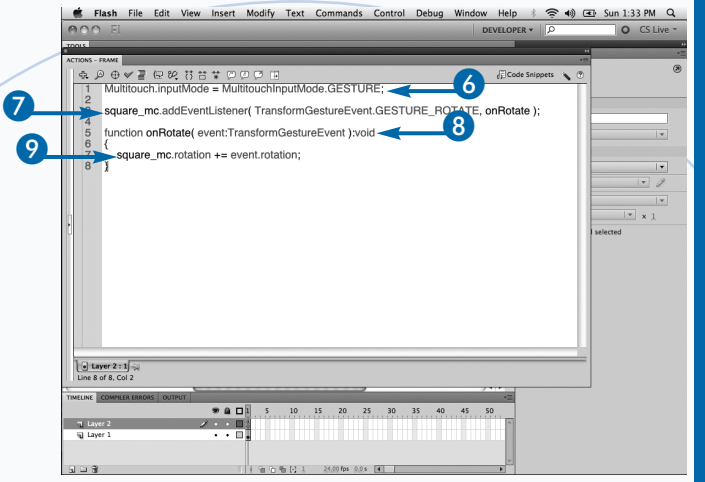


07_904329-ch05.indd 84 8/26/10 10:27 PM

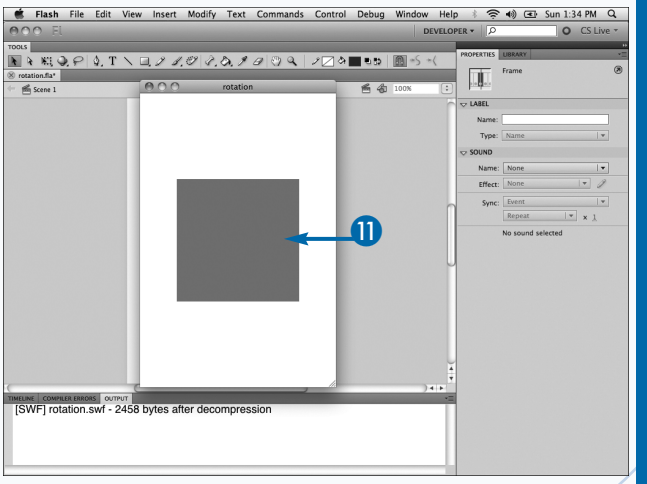
6. Set the input mode, such as `Multitouch.inputMode = MultitouchInputMode.GESTURE;`.
7. Add a listener for the rotate gesture, such as `square_mc.addEventListener( TransformGestureEvent.GESTURE_ROTATE, onRotate );`.
8. Create an event handler for your rotate listener.

Note: *See Chapter 2 for more details on this step.*

9. Set the new rotation, such as `square_mc.rotation += event.rotation;`.



10. Publish and install the application on your device.
11. Place two fingers on the square and rotate one around the other.



Apply It

If you are feeling adventurous and want to try and program this gesture yourself, you can do so by tracking two fingers. One way to determine the new rotation of your object is to find the angle between two lines. To do this, you will need to store the previous location of the two touches. Those two points will form line 1, and the current touch points will form line 2. Here is a method that calculates the angle in degrees from two lines:

```
function angleBetweenLines( line1Start:Point,  line1End:Point,  line2Start:Point,
  line2End:Point):Number{
var a:Number = line1End.x - line1Start.x;
var b:Number = line1End.y - line1Start.y;
var c:Number = line2End.x - line2Start.x;
var d:Number = line2End.y - line2Start.y;
var degs:Number = Math.cos(((a*c) + (b*d)) / ((Math.sqrt(a*a + b*b)) * (Math.sqrt(c*c +
  d*d))));
return degs * ( 180 / Math.PI );
}
```


07_904329-ch05.indd 85 8/26/10 10:27 PM

Respond to Pan Events

Because of the limited screen real estate, there will be times when your content will be bigger than the viewing area. When this happens, you will want to give the users the ability to scroll or pan the content so that they can see any hidden content. This can also happen when you zoom in on objects, and objects scale up past the bounds of the screen. For example, consider a photo gallery application. When you initially display the image, you would show it at a size that fills the screen. But you may allow the user to zoom into specific areas of the image using the `TransformGestureEvent.GESTURE_ZOOM` event. The user scales the image up twice its original size, and now the part of the image that she is interested in is off the screen. You will want to let her pan the image around in order to see any areas that were off-screen.

To detect for a pan gesture, you can listen for the `TransformGestureEvent.GESTURE_PAN` event on any `InteractiveObject`. This event is fired when it is detected that the user has placed two fingers on the object and is dragging them around the screen. Only one finger must be on the targeted object, and the other controls the direction in which to pan the object. The `TransformGestureEvent` object that gets passed to the event handler contains two properties that you will use to move your object. `TransformGestureEvent.offsetX` and `TransformGestureEvent.offsetY` give the difference in position since the last pan event. To move the targeted object, add the current x and y values of your object to these values.

Respond to Pan Events

Set Panning on an Image

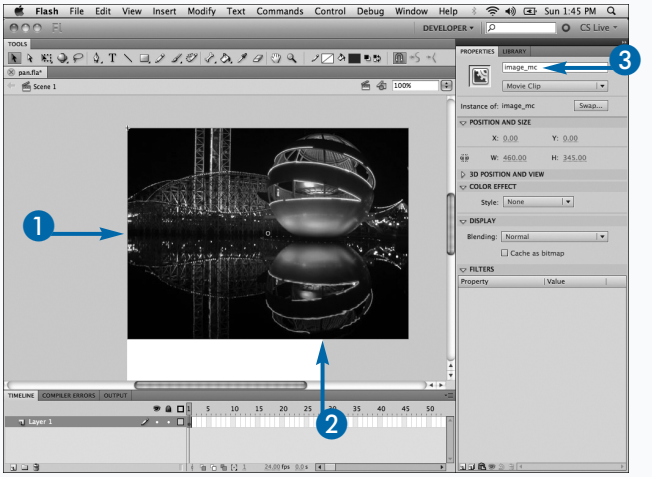
1. Import an image onto the Stage.

Note: *See Chapter 6, "Working with Images," for more details.*

2. Convert the image to a `MovieClip`.

Note: *See Chapter 2 for more details on this topic.*

3. Give the `MovieClip` an instance name, such as `image_mc`.

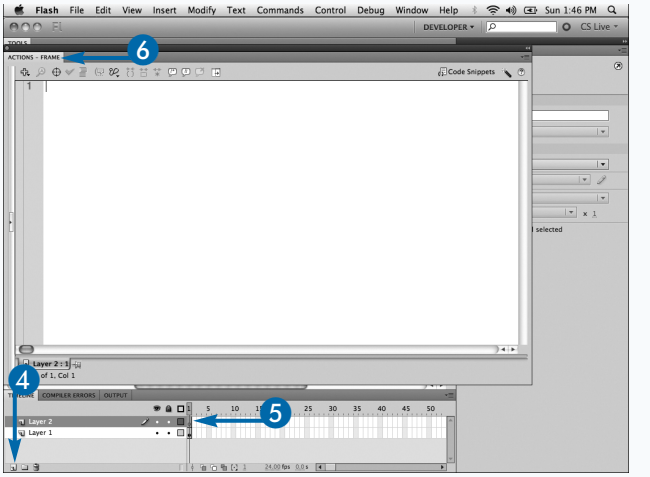


4. In the Timeline panel, click the New Layer button.

   A new layer is created.

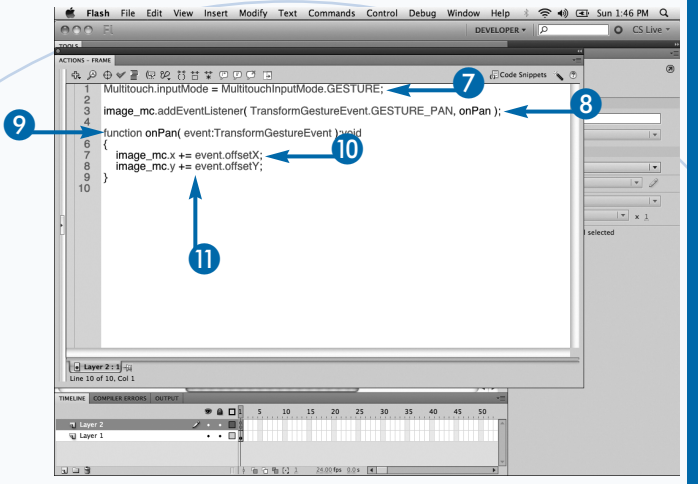
5. Select the new layer.

6. Open the Actions panel.



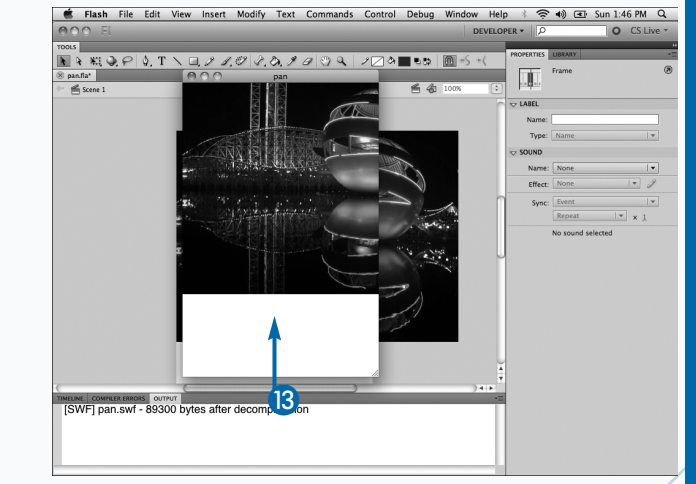
07_904329-ch05.indd 86 8/26/10 10:27 PM

7. Set the input mode, such as `Multitouch.inputMode = MultitouchInputMode.GESTURE;`.
8. Add a listener for the pan gesture, such as `image_mc.addEventListener( TransformGestureEvent.GESTURE_PAN, onPan );`.
9. Create an event handler for your pan listener.
10. Set the new `x` position based on the gesture, such as `image_mc.x += event.offsetX;`.
11. Set the new `y` position based on the gesture, such as `image_mc.y += event.offsetY;`.



Test the Application

12. Publish and install the application on your device.
13. Place two fingers on the image and move them around the screen to pan the image.



Apply It

There are other ways to pan objects than using the `TransformGestureEvent.GESTURE_PAN` event. For example, you could use `TouchEvent.TOUCH_MOVE` to drag the object around if you wanted to, as in the following:

```
Multitouch.inputMode = MultitouchInputMode.TOUCH_POINT;
this.stage.addEventListener(TouchEvent.TOUCH_MOVE, onTouchMove);
function onTouchMove( event:TouchEvent ):void{
image_mc.x = event.stageX;
image_mc.y = event.stageY;
}
```

However, you cannot detect for gestures and touches at the same time in your application. So if want to detect for any other gestures and want the ability to pan and drag, you will need to use the `TransformGestureEvent.GESTURE_PAN` event.

07_904329-ch05.indd 87 8/26/10 10:27 PM

Respond to Swipe Events

The swipe is a relatively simple gesture. It can be initiated by swiping your finger either vertically or horizontally in a straight line with a single finger. This gesture is most commonly used to navigate through a set of content. A good example of this is the Gallery application, which uses swipe gestures to navigate through the photos on your device.

You can listen for a swipe gesture on any `InteractiveObject` by adding listening for its `TransformGestureEvent.GESTURE_SWIPE` event. When a swipe gesture is detected, a `TransformGestureEvent` object is passed as an argument to your event handler. The `offsetX` property will return either 1 for a swipe right or -1 for a swipe left. The `offsetY` property will return either 1 for a swipe down or -1 for a swipe up.

As soon as you know which way the swipe occurred, you can adjust your content accordingly.

The example below shows importing two images to the Stage and moving them left or right depending on the direction of the swipe. When you first load the application, you are going to place the first image on the Stage and have it take up the entire screen area. Because it is the first image, you are going to only allow for left swipes. If you swiped the content right, there would not be an image to move into its place. When you swipe left, you are going to animate the current image off-screen to the left and animate the second image on-screen from the right. When the second image is shown, you are going to check for right swipes. When a right swipe is detected, you are going to animate the second image off-screen to the right and animate the first image in from the left.

Respond to Swipe Events

Set Swiping for Two Images

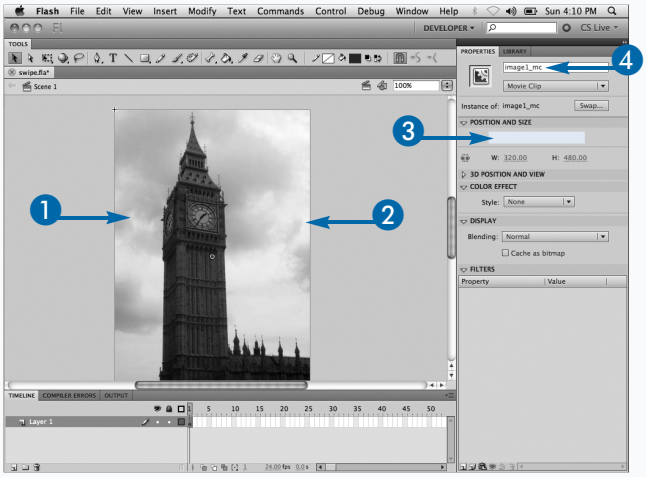
1. Import an image onto the Stage.

Note: *See Chapter 6 for more details.*

2. Convert the image to a `MovieClip`.

Note: *See Chapter 2 for more details.*

3. Place the image at 0,0.
4. Give it an instance name, such as `image1_mc`.



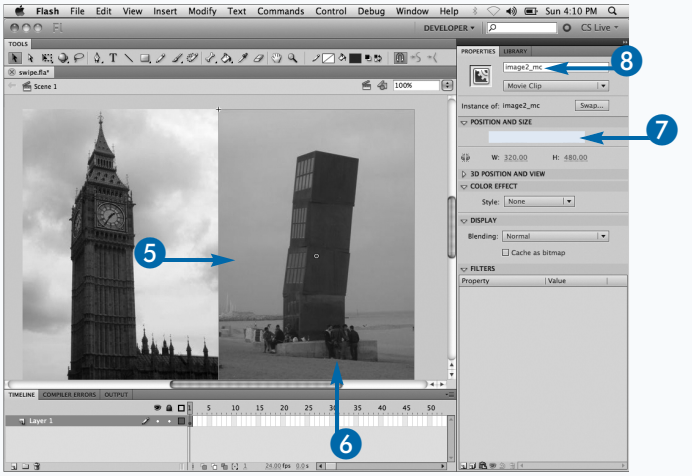
5. Import a second image onto the Stage.

Note: *See Chapter 6 for more details.*

6. Convert the image to a `MovieClip`.

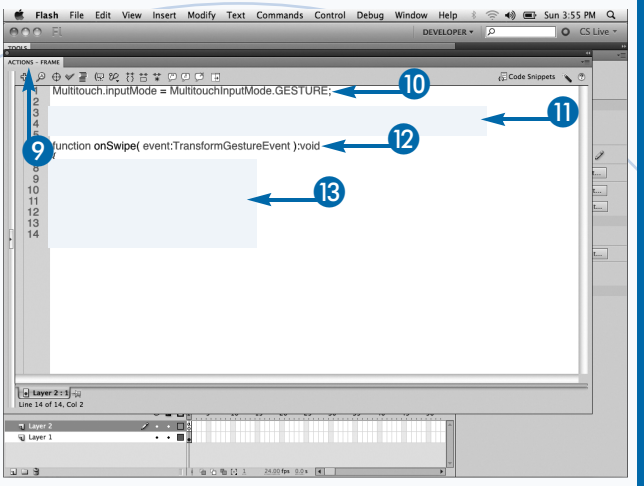
Note: *See Chapter 2 for more details.*

7. Place the image at 320,0.
8. Give it an instance name, such as `image2_mc`.

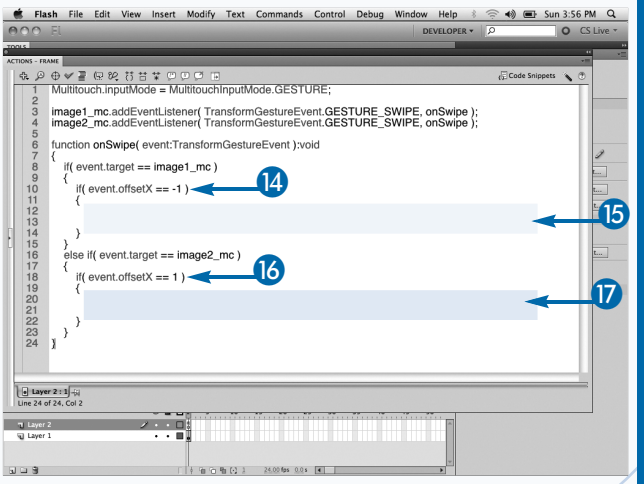


07_904329-ch05.indd 88 8/26/10 10:27 PM

9. Open the Actions panel.
10. Set the input mode, such as `Multitouch.inputMode = MultitouchInputMode.GESTURE;`.
11. Add a swipe gesture listener to both images.
12. Create an event handler, such as `onSwipe`.
13. Check to see which image detected the swipe gesture.



14. Check to see if it was a left swipe.
15. If so, animate the first image to -320 x and the second image to 0 x.
16. Check to see if it was a right swipe.
17. If so, animate the first image to 0 x and the second image to 320 x.



Test the Application

18. Publish and install the application on your device.
19. Swipe the images left and then swipe them right.

Apply It

The way shown in this section is the easiest way to detect for swipes, but you can also detect touches independently and determine which direction a swipe is going. By doing this, you have more control over how fast and how far the swipe has to go before it is classified as a swipe:

```
image1_mc.addEventListener( TouchEvent.TOUCH_BEGIN, onBegin );
image1_mc.addEventListener( TouchEvent.TOUCH_MOVE, onMove );
var startX:Number;
var startY:Number;
function onBegin( event:TouchEvent ):void{
    startX = event.stageX;
    startY = event.stageY;
}
function onMove( event:TouchEvent ):void{
if( startX > event.stageX ){
            // left swipe detected
        }else{
            //right swipe detected
        }
}
```


07_904329-ch05.indd 89 8/26/10 10:27 PM

Listen for Accelerometer Events

A popular interaction with mobile devices, especially among games, is the use of the accelerometer. Driving games use the accelerometer, allowing the users to rotate their devices back and forth in order to steer their vehicle. Also, shaking the device has become a popular way to incorporate Easter eggs into your game or application.

The new `Accelerometer` class lets you receive acceleration data from the on-board accelerometer chip of the device. As the device moves, you will receive linear acceleration data along the x, y, and z axes. To receive this data, listen for the `AccelerometerEvent.UPDATE` event on the `Accelerometer` object. An `AccelerometerEvent` object will be passed to your event handler, which contains `accelerationX`, `accelerationY`, and `accelerationZ` properties, a value for each axis.

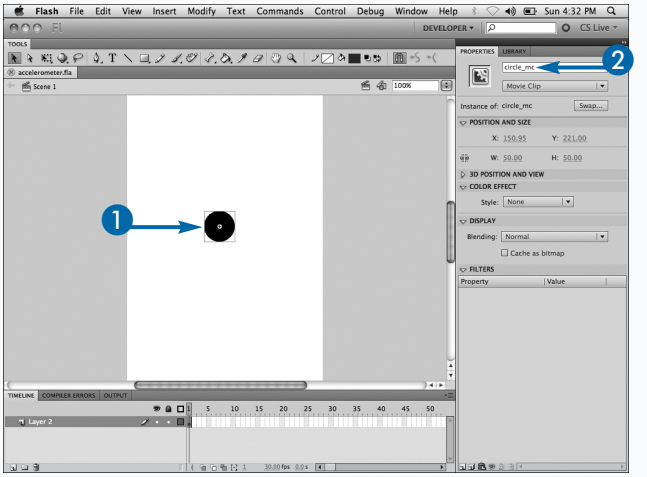
The `accelerationX` property represents the acceleration measured in Gs along the x-axis. This axis runs from the left to the right of the device when it is in its upright position. The `accelerationY` property represents the acceleration measured in Gs along the y-axis. This axis runs from the bottom of the phone to the top. In the case of an Android, this axis runs from the earpiece to the bottom of the device. The `accelerationZ` property represents the acceleration measured in Gs along the z-axis. The axis runs perpendicular to the face of the phone, and the value will be positive as it moves closer to you.

The example below shows moving a ball around the screen as the device is tilted and moved around.

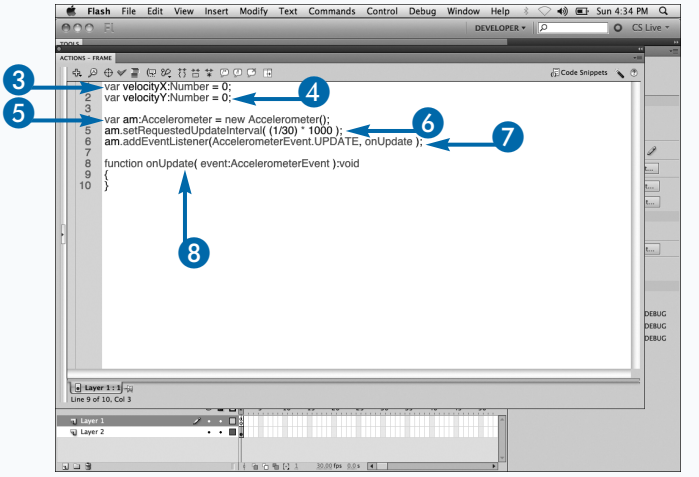
Listen for Accelerometer Events

Create a Ball That Moves with the Device

1. Create a circle `MovieClip`.
2. Give it an instance name, such as `circle_mc`.

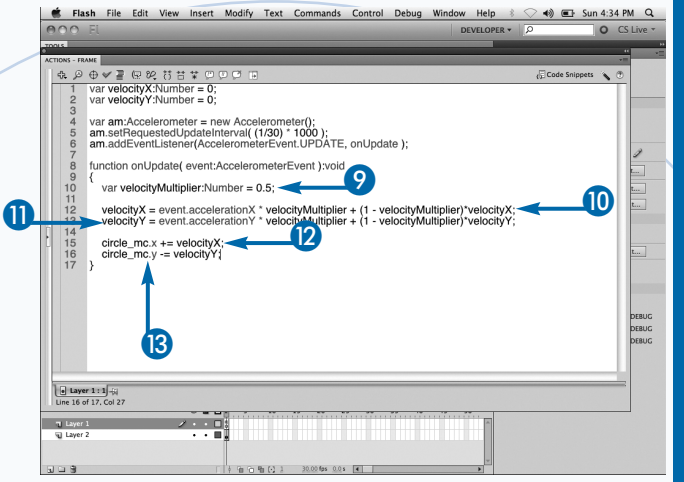


3. Create a `velocityX` variable, such as `var velocityX:Number = 0;`.
4. Create a `velocityY` variable, such as `var velocityY:Number = 0;`.
5. Create an `Accelerometer` variable, such as `var am:Accelerometer = new Accelerometer();`.
6. Set the update interval of the accelerometer.
7. Add an update listener to the accelerometer.
8. Create an event handler, such as `onUpdate`.



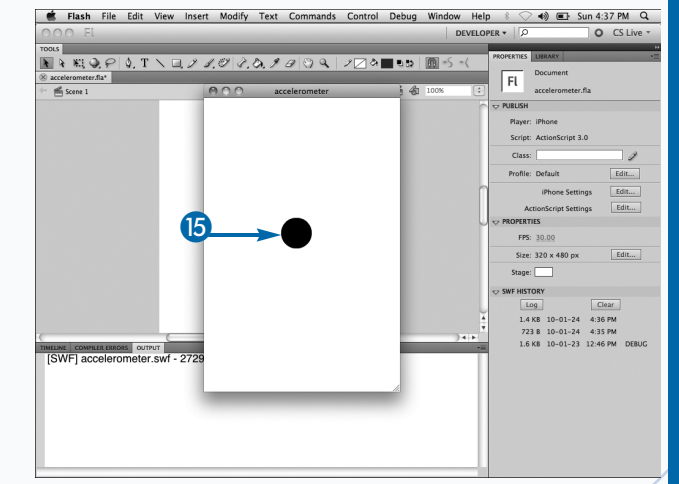
07_904329-ch05.indd 90 8/26/10 10:27 PM

9. Create a `Number` variable, such as `var velocityMultiplier: Number = 0.5;`.
10. Apply the `accelerationX` value to calculate the velocity of `x`.
11. Apply the `accelerationY` value to calculate the velocity of `y`.
12. Set the `x` property of the circle instance, such as `circle_mc.x += velocityX`.
13. Set the `y` property of the circle instance, such as `circle_mc.y -= velocityY`.



Test the Application

14. Publish and install the application on your device.
15. Move the device around to move the circle on the screen.



Apply It

One thing to keep in mind when trying to animate an object with values from the accelerometer is that you are not guaranteed to get values at a regular interval. So if you are trying to create nice, smooth, time-based animations based on that data, you will need to store the time between each accelerometer update and factor that into your equation.

Another option is to animate the object on a `Timer` or `EnterFrame` event and only update the velocities in the accelerometer update handler. To do so, remove the following code from the `onUpdate` method:

```
circle_mc.x += velocityX;
circle_mc.y -= velocityY;
```

And add the following code to the example:

```
stage.addEventListener( Event.ENTER_FRAME, onFrame );
function onFrame( event:Event ):void{
    circle_mc.x += velocityX;
    circle_mc.y -= velocityY;
}
```


07_904329-ch05.indd 91 8/26/10 10:27 PM

Determine If the Accelerometer Is Available

When developing an application that takes advantage of the accelerometer, you are going to need to place the application on your device in order to test it. This can slow down development if every time you make changes to your application, you need to test it on the device. This is extremely inefficient, especially if you are not testing the portion of your application that uses the accelerometer. To get around this issue, a good practice is to detect to see if the accelerometer is available and give yourself another way of interacting with your application if it is not.

For example, if you need to detect for a shake motion, you could place a button on the screen that when pressed would simulate a shake. Doing this allows you to test on your computer, and when you are satisfied that everything is working, you can put your application on the device to test your accelerometer code.

The `Accelerometer` class has a static property on it named `isSupported` to detect if it is available. It is a read-only property that is set to true if it is available; otherwise, it is false.

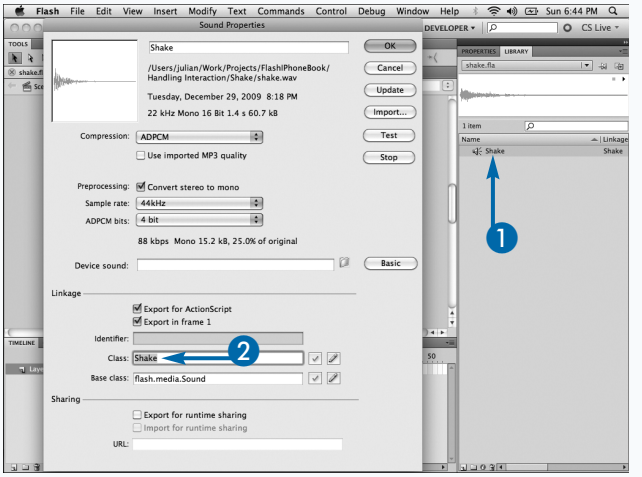
The example below shows comparing the current acceleration data to the previous to see if there has been a big enough shake. If a shake has occurred, you are going to play a sound. To detect for the shake, compare the previous acceleration data of all three axes to the current data. If there was enough of a difference in acceleration in any direction, you can determine that the user has shaken the device.

Determine If the Accelerometer Is Available

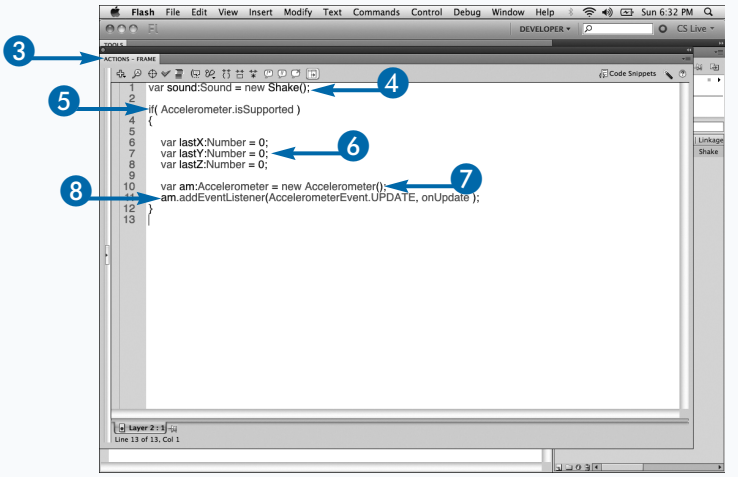
Respond to a Shake with a Sound

1. Import a sound effect audio file.
2. Give the sound a class name, such as `Shake`.

Note: *For more details on importing sounds, check out Chapter 7, "Working with Sound."*



3. Open the Actions panel.
4. Create a `Sound` variable for the shake sound, such as `var sound:Sound = new Shake();`.
5. Check to see if the accelerometer is available.
6. Create `Number` variables for the previous x, y, and z positions, such as `lastX`, `lastY`, and `lastZ`.
7. Create a new `Accelerometer` variable, such as `var am:Accelerometer = new Accelerometer();`.
8. Add an update listener to the `Accelerometer`.

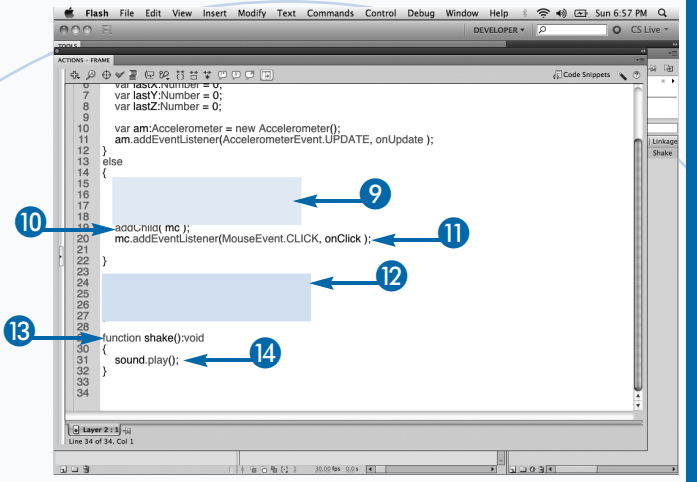


07_904329-ch05.indd 92 8/26/10 10:27 PM

9. Create a `Sprite` and draw a rectangle in it.

Note: *See Chapter 2.*

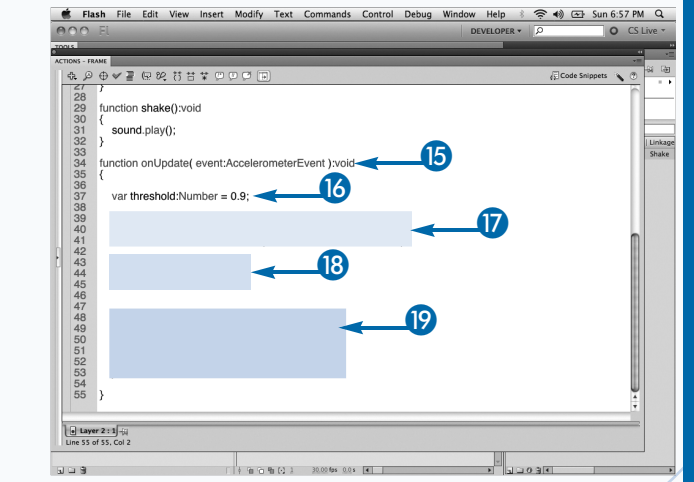
10. Add the sprite to the Stage.
11. Add a listener for the mouse click.
12. Create an event handler, such as `onClick`, and call the `shake` method inside it.
13. Create a `shake` function.
14. Play the sound, such as `sound.play();`.



15. Create an event handler, such as `onUpdate`.
16. Create a `Number` variable, such as `var threshold:Number = 0.9;`.
17. Calculate the difference between the current data and the previous data.
18. Store the current acceleration for the next update.
19. Determine if there was a big enough difference in any of the axes and call the `shake` method.

Test the Application

20. Publish the application and run it on your device and the desktop to see the difference.



Apply It

There is also an `Accelerometer.muted` property that you can use to detect if the accelerometer is available. This is used to determine if the user has denied access to the accelerometer. Currently, Android devices do not have an option that allows the user to disable the accelerometer, but one may exist on other platforms. So if you are targeting multiple platforms, you may want to check this property as well for determining if the accelerometer is available. If you are just building Android applications, you do not have to worry about it. Here is how to use this property:

```
if( Accelerometer.isSupported || !Accelerometer.muted )
{
//Accelerometer is available.
}
```


07_904329-ch05.indd 93 8/26/10 10:27 PM

Determine Device Orientation

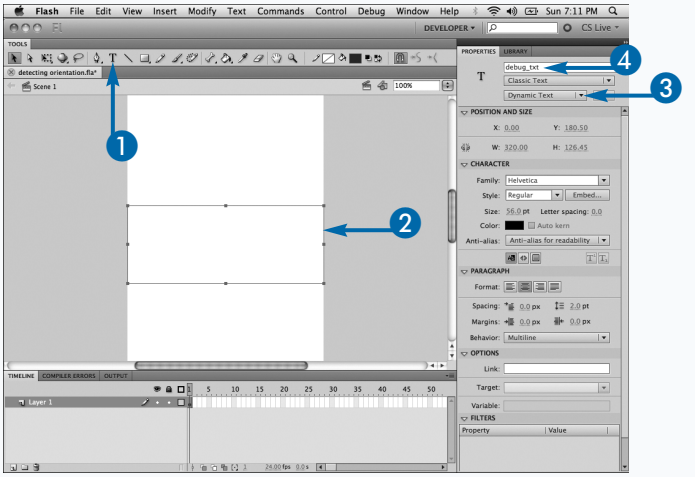
Determining the orientation of the device can be easily achieved by reading the acceleration data from the accelerometer. You can detect to see if the device is positioned on any one of its six sides. This can be used in order to rotate content so that it matches the rotation of the device. It can also be used to detect user input. For example, say that you wanted the users to flip an object in a game. You could have the users actually flip their devices in order to flip the object in the game. Keep in mind, though, that the more extreme the gesture, the bigger the chance the users can have their devices flying out of their hands. And nobody will be doing that gesture more than you during development.

After you are familiar with reading the acceleration data from the accelerometer, determining the orientation is pretty simple. The following example references the Nexus One and most other Android phones. If `accelerationX` is greater than 0.5, then the device is laying on its right side, the side opposite the volume controls. If `accelerationX` is less than -0.5, the device is laying on its left side, the side with the volume controls. If `accelerationY` is greater than 0.5, the device is laying on its top side, the side with the headphone jack and lock button. If `accelerationY` is less than -0.5, the device is standing up, with the side with the connector jack pointing to the floor. If `accelerationZ` is greater than 0.5, the device is lying with its screen down. If `accelerationZ` is less than -0.5, the device is lying with its back down.

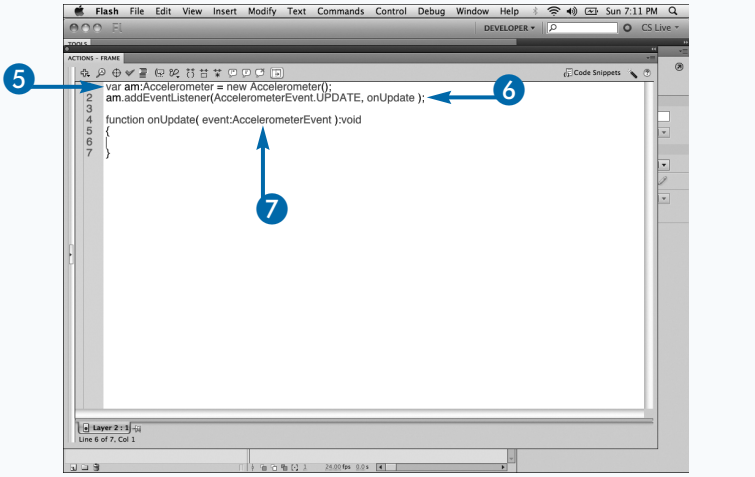
Determine Device Orientation

Create a TextField That Displays the Device Orientation

1. Click the Text tool.
2. Create a `TextField` on the Stage.
3. Click here and select Dynamic Text as the `TextField`'s type.
4. Give the text field an instance name, such as `debug_txt`.

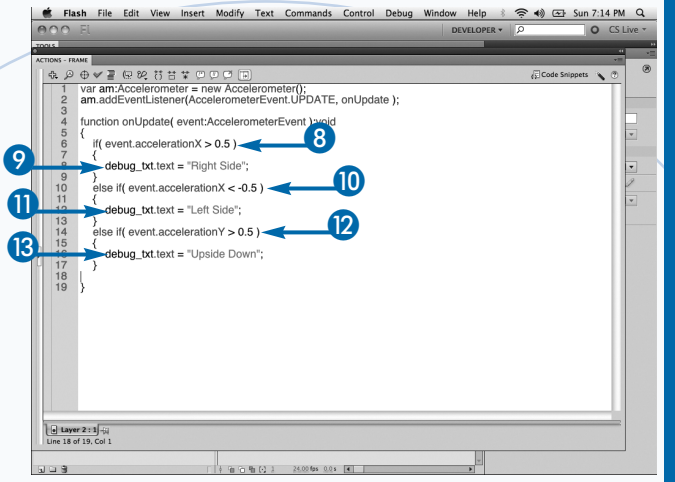


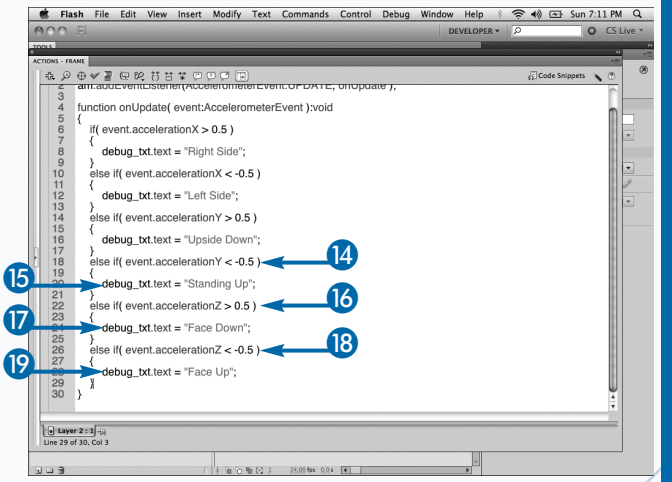
5. Create a new `Accelerometer` variable, such as `var am:Accelerometer = new Accelerometer();`.
6. Add an update event listener to the accelerometer variable.
7. Create an event handler, such as `onUpdate`.



07_904329-ch05.indd 94 8/26/10 10:27 PM

8. Check to see if `accelerationX` is greater than 0.5.
9. Set the text field, such as `debug_txt.text = "Right Side";`.
10. Check to see if `accelerationX` is less than -0.5.
11. Set the text field, such as `debug_txt.text = "Left Side";`.
12. Check to see if `accelerationY` is greater than 0.5.
13. Set the text field, such as `debug_txt.text = "Upside Down";`.
14. Check to see if `accelerationY` is less than -0.5.
15. Set the text field, such as `debug_txt.text = "Standing Up";`.
16. Check to see if `accelerationZ` is greater than 0.5.
17. Set the text field, such as `debug_txt.text = "Face Down";`.
18. Check to see if `accelerationZ` is less than -0.5.
19. Set the text field, such as `debug_txt.text = "Face Up";`.





Test the Application

20. Publish and install the application on your device. Change the orientation.

Extra

After you have determined the orientation of the device, you will want to rotate your content for your user. If the device is lying on its left side, you will want to rotate your content 90 degrees. If the device is lying on its right side, rotate your content -90 or 270 degrees. If the device is upside down or lying on its top, rotate your content 180 degrees. For the other three sides, the content's rotation should be set to 0 degrees. One thing to keep in mind when rotating content this way is that you will have to reposition it as well. You could also have separate views for each rotation and swap them as the device rotates. The advantage to this is that you can potentially show different sized graphics when the device is rotated to either its left or right side to maximize the change in screen real estate. There is no right way to do it, and you will have to decide which way works best for you on a case-by-case basis.

07_904329-ch05.indd 95 8/26/10 10:27 PM

Detect Which Way Is Up

When you are developing applications that allow the user to rotate the device, it will be important for you to make sure that your content is rotated and displayed properly. You could implement something similar to the example in the section "Determine Device Orientation," or you could detect which side is pointing up. Detecting which side is pointing up allows you to rotate your content to a more precise angle, instead of just every 45 degrees.

First, set up the `Accelerometer` so that you can receive the update events. If you are unsure of how to do this, have a look at the example in the section "Listen for Accelerometer Events." After you have an event handler created and receiving acceleration data, you can use the `Math.atan2` method to calculate the angle in radians.

The `Math.atan2` method takes two arguments, `y` and `x`. It is important to note that the `y` property is always the first argument, which is usually different in any other method that accepts `x` and `y` properties as arguments. To calculate the angle, you pass in the `accelerationY` value and the negative value of `accelerationX`. Because a `DisplayObjects.rotation` property is expecting an angle in degrees, you will need to convert the radians to degrees in order to set the rotation of your content. The following equation shows you how to do the conversion:

```
var degrees:Number = radians * ( 180 / Math.PI
 );
```

After you have the angle in degrees, you can set it to the rotation property of your object. Below is a very simple example that rotates an image of an arrow to which way points up.

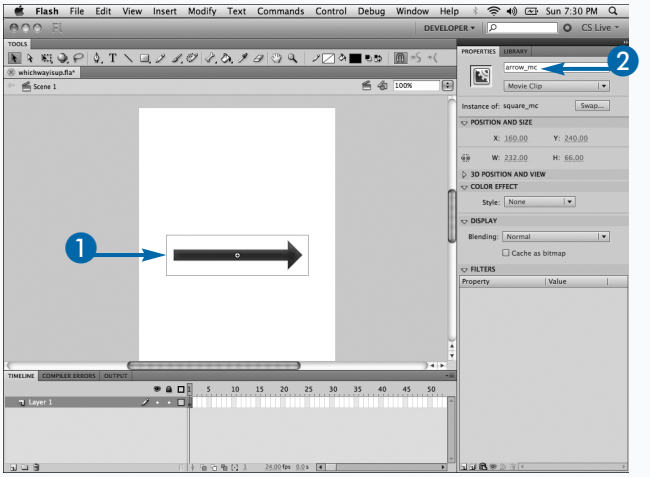
Detect Which Way Is Up

Create an Arrow to Point Up

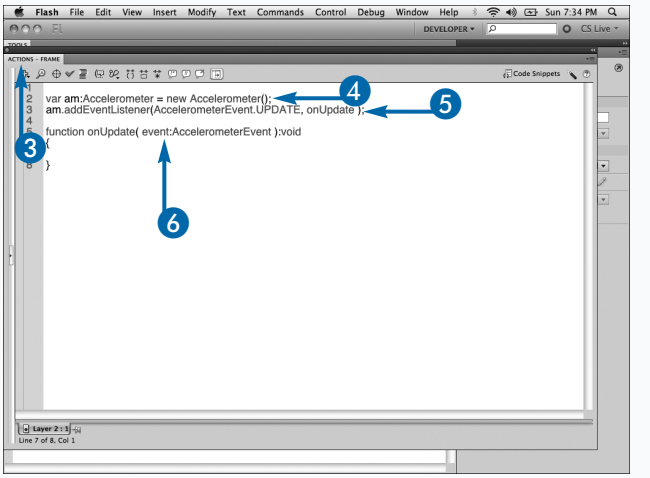
1. Import an arrow graphic and convert it to a `MovieClip`.

Note: *See Chapter 2 for more details on this topic.*

2. Give it an instance name, such as `arrow_mc`.

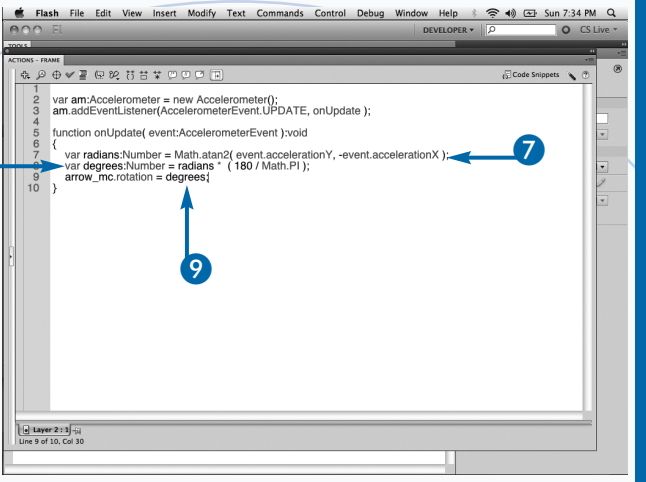


3. Open the Actions panel.
4. Create a new `Accelerometer` variable, such as `var am:Accelerometer = new Accelerometer();`.
5. Add an update event listener to the accelerometer variable.
6. Create an event handler, such as `onUpdate`.



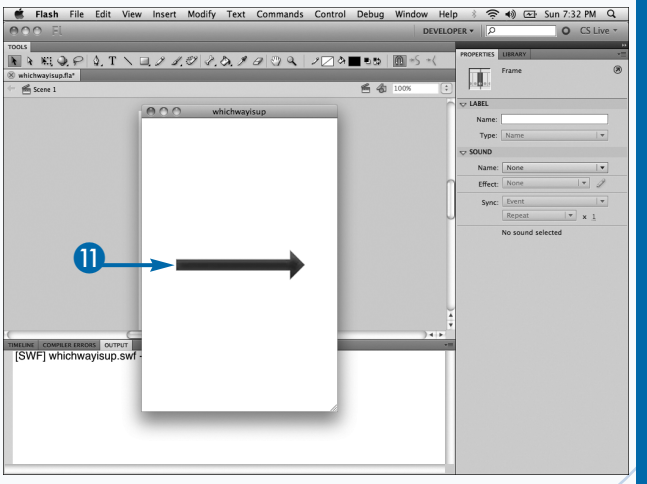
07_904329-ch05.indd 96 8/26/10 10:27 PM

7. Calculate the radians based on the acceleration values.
8. Convert the radians to degrees.
9. Set the rotation, such as `arrow_mc.rotation = degrees;`.



Test the Application

10. Publish and install the application on your device.
11. Rotate the device around to rotate the arrow.



Apply It

Changing the example so that the arrow is pointing to the side that is down is relatively easy. You simply need to change the values that you pass in to the `Math.atan2` function to be negative `accelerationY` and positive `accelerationX`. This can be used to make sure that objects always fall the direction that gravity is pulling them. Here is what the function looks like with these changes:

```
var radians:Number = Math.atan2( -event.accelerationY, event.accelerationX );
```


07_904329-ch05.indd 97 8/26/10 10:27 PM

Filter Accelerometer Data

If you have been going through the other accelerometer examples or experimenting on your own, you have probably noticed that the data can sometimes contain some noise. Depending on how you are visualizing the data, this noise may cause your graphics to jump around quite a bit. To reduce the noise, you will want to smooth out the values from the `Accelerometer` by filtering out unwanted values. Smoothing data sets is used to capture patterns in the data while removing any noise. This technique is often used in analyzing images and sound waves.

The `accelerationZ` property in the `AccelerometerEvent` object represents gravity. To better understand this, place your device on a flat surface, like a table. You should see that the `accelerationX` and `accelerationY` properties are approximately 0, and the `accelerationZ` property should be approximately -1.

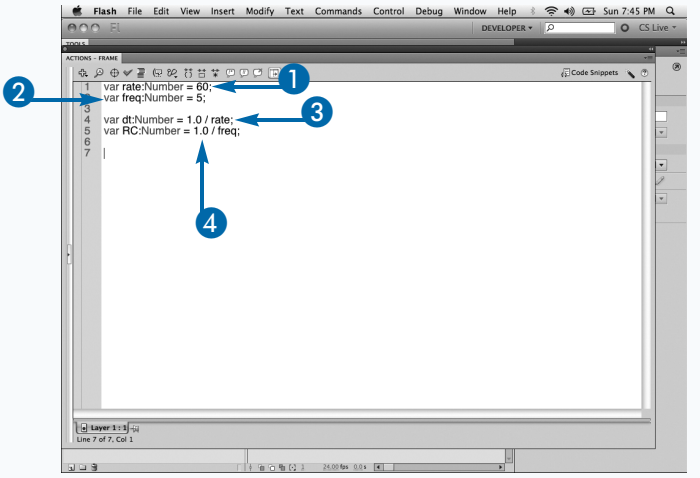
To remove the effects of gravity, you can use a *high-pass filter,* which reduces the amplitude of the cutoff frequency. This filter reduces some of the noise and gives you smoother results between updates.

The `dt` variable in the example below represents a time interval, and the `RC` variable represents a time constant. These two variables are used to calculate the `filterConstant` variable. A large `filterConstant` variable suggests that the data will decay very slowly over time and will also be strongly influenced by small changes in the accelerometer. A small `filterContstant` variable suggests that the data will decay quickly and will require large changes in the accelerometer in order to change the output. Changing the `rate` and `freq` variables in this example gives you more control over how you would like your acceleration data to be filtered.

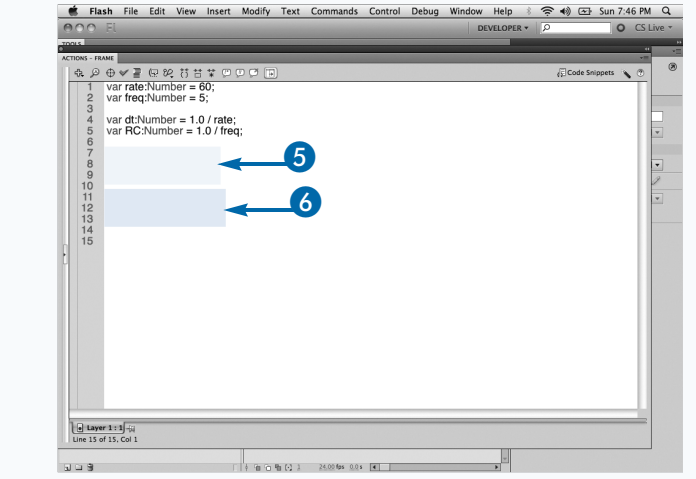
Filter Accelerometer Data

Using a High-Pass Filter

1. Create a `Number` variable, such as `var rate:Number = 60;`.
2. Create a second `Number` variable, such as `var freq:Number = 5;`.
3. Create a third `Number` variable, such as `var dt:Number = 1.0/rate;`.
4. Create a fourth `Number` variable, such as `var RC:Number = 1.0/freq;`.

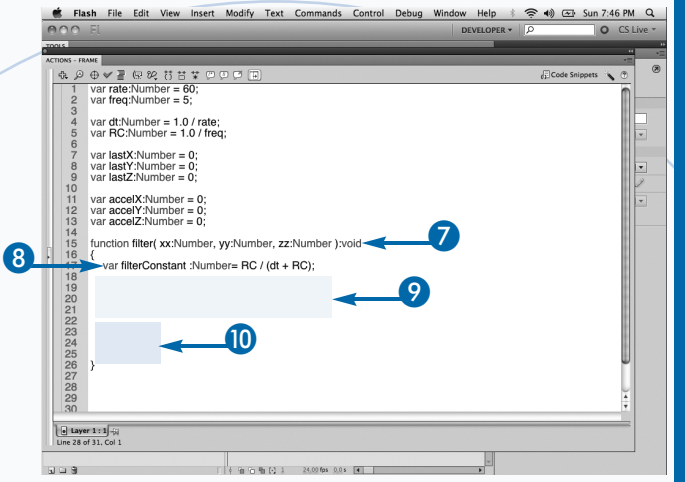


5. Create `Number` variables to store the acceleration data.
6. Create `Number` variables to store the filtered acceleration values.

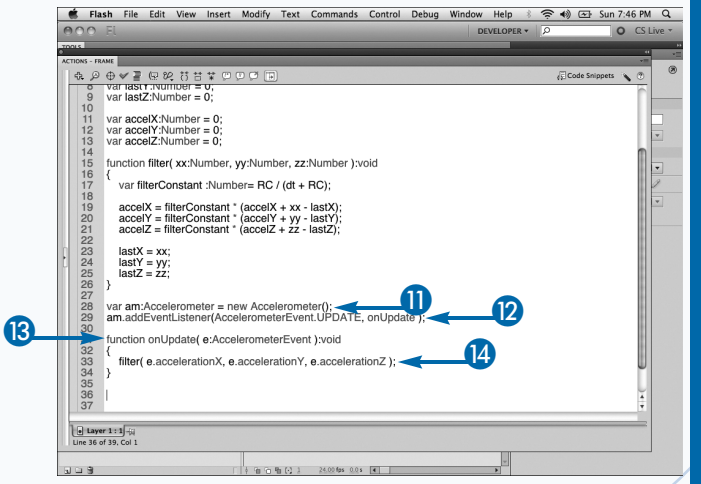


07_904329-ch05.indd 98 8/26/10 10:27 PM

7. Create a `filter` function.
8. Create a `Number` variable, such as `var filterConstant :Number= RC / (dt + RC);`.
9. Filter the data for all three axes.
10. Store the current acceleration values for the next update.



11. Create a new `Accelerometer` variable, such as `var am:Accelerometer = new Accelerometer();`.
12. Add an update event listener to the accelerometer variable.
13. Create an event handler, such as `onUpdate`.
14. Call the `filter` function.



Publish the Application

15. Publish and install the application on your device.

Apply It

You can also experiment with using a low-pass filter instead of a high-pass one. A *low-pass filter* reduces the amplitude of frequencies higher than the frequency cutoff and isolates the effects of gravity. To use this type of filter, you can simply replace the code in the `filter` method in the example shown here with the following:

```
var filterConstant:Number = dt / (dt + RC);
accelX = xx * filterConstant + accelX * (1.0 - filterConstant);
accelY = yy * filterConstant + accelY * (1.0 - filterConstant);
accelZ = zz * filterConstant + accelZ * (1.0 - filterConstant);
```

I encourage you to experiment with both filters and see which one works best for your specific implementation.

07_904329-ch05.indd 99 8/26/10 10:27 PM