CISC 110 Week 11

Extra Features for Optional Use in Projects
Today

- Shape Tweens
- Color Transforms
- Glow Filters
- Sound Control
- Input Text Fields and Focus Events
- The String Class
- Drawing Vectors Dynamically
Next Week

- Class time will be allocated to work on projects
  - Potentially in Lab room, stay tuned
Login in

- Load exercises for this week
Shape Tweens

Steps:

- Create any simple filled shape
- Insert a blank key frame on a later frame and create a different shape
- Go to the first frame and select Insert | Shape Tween
ColorTransform Class

Purpose: to change a MovieClip instance’s color dynamically

Steps:
1. Import flash.geom.ColorTransform in your .as file
2. Create a new ColorTransform object
3. Set its red, blue, and green offsets
4. Assign the ColorTransform object to your MovieClip object
5. Add your MovieClip to the stage with the addChild method
ColorTransform Class

var veggie: Pepper = new Pepper( );

var veggieColor = new ColorTransform( );
veggieColor.redOffset = 0;
veggieColor.greenOffset = 255;  // Now green
veggieColor.blueOffset = 0;
veggie.transform.colorTransform = veggieColor;

addChild( veggie );
Glow Filters

// Create a new GlowFilter object
var gf: GlowFilter = new GlowFilter();

// Set blur width to 5 pixels and height to 2 pixels
gf.blurX = 5;   gf.blurY = 2;

// Set color to red: red = 255; blue = 0; green = 0
gf.color = 0xff0000;  // in hexadecimal

// Set center knocked out; shows what’s behind it
gf.knockout = true;
Glow Filters

Add a GlowFilter within an array to a MovieClip. You could put many different kinds of filters in an array and add them to a MovieClip object

// Add the GlowFilter gf to the MovieClip sun
sun.filters = [ gf ];
GlowFilters

The properties of a GlowFilter object can be changed and then added again within an array to a MovieClip object

// Increase the width and height of the blur
// Do not show what’s behind the center

gf.blurY += 0.1;
gf.blurX += 0.5;
gf.knockout = false;
sun.filters = [gf];
Sound Control Buttons

**Purpose:** to let the user turn music/sounds on or off

**Steps:**
1. Import `flash.media.SoundChannel` in your .as file
2. Import the sound file into your library and create a linkage to make it a custom sound class
3. Create an instance of the your custom sound class
4. Create a new `SoundChannel` object to control it
5. Use one or more buttons to turn the sound on or off via the play method in the Sound class and the stop method in the SoundChannel class
Sound Control Buttons

// Global Variables:
// Create an instance of your custom sound class
var gameMusic: MyMusic = new MyMusic();
// Create a new SoundChannel object to control it
var gameMusicControl: SoundChannel = new SoundChannel();

// Within Constructor or other function:
// Listeners for two buttons to turn the sound on or off
onBtn.addEventListener(MouseEvent.CLICK, turnMusicOn);
offBtn.addEventListener(MouseEvent.CLICK, turnMusicOff);
Sound Control Buttons

// Button Handlers: turn sound on or off via play method in Sound class and stop method in SoundChannel class

function turnMusicOn( evt: MouseEvent )
{
    gameMusicControl = gameMusic.play( );
}

function turnMusicOff( evt: MouseEvent )
{
    gameMusicControl.stop( );
}
Input Text Fields

**Purpose:** To allow user input

Using the text tool in Flash, we create a TextField. In the Properties Panel, we set its text type to be Input Text and its other properties as desired, embed the font, and give it an instance name: msgBox.

We can then look at text that the user types in it:

```
trace( msgBox.text );
```

But how do we know when the user has typed text in it?
1. Use a Button

This is the method we’ve used so far:

Use a Button that the user must press after they’ve filled in a text box in order for another action to take place.
2. Use a KeyboardEvent

// Add event listener for any key pressed
stage.addEventListener (KeyboardEvent.KEY_DOWN, readMsg);

// Add event handler that only prints message after
// user types the Enter key
// function readMsg( evt: KeyboardEvent ) : void

function readMsg( evt: KeyboardEvent ) : void
{
    if( evt.keyCode == Keyboard.ENTER )
        trace( msgBox.text );
}

**Note:** May only work correctly outside the Flash environment
3. Use a FocusEvent

FOCUS_OUT occurs when user deselected an object: i.e., clicks anywhere outside of the object with mouse

// Add event listener for user deselecting text field
msgBox.addEventListener(FocusEvent.FOCUS_OUT, readMsg);

// Add event handler that then prints the message
function readMsg( evt: FocusEvent ) : void
{
    trace( msgBox.text );
}

FOCUS_IN Event

FOCUS_IN occurs when user selects an object

// Add event listener for user selecting text field
msgBox.addEventListener( FocusEvent.FOCUS_IN, blankOutText );

// Add event handler that sets input text field to blank
function blankOutText( evt: FocusEvent ): void
{
    msgBox.text = " ";
}
### The String Class

<table>
<thead>
<tr>
<th>ActionScript Statement</th>
<th>Value of Strings</th>
</tr>
</thead>
<tbody>
<tr>
<td>var myName: String;</td>
<td>None yet</td>
</tr>
<tr>
<td>var yourName: String;</td>
<td>None yet</td>
</tr>
<tr>
<td>myName = “Jimena”;;</td>
<td>“Jimena”</td>
</tr>
<tr>
<td>yourName = myName;</td>
<td>“Jimena”</td>
</tr>
</tbody>
</table>

**Note:** Strings are indexed starting at 0, just like arrays.
## Special Characters in Strings

<table>
<thead>
<tr>
<th>Character</th>
<th>Escape Char Code</th>
<th>Example String</th>
<th>What is Displayed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tab</td>
<td>\t</td>
<td>“Carlos \t Santana”</td>
<td>Carlos Santana</td>
</tr>
<tr>
<td>Line Break</td>
<td>\n</td>
<td>“Dixie \n Chicks”</td>
<td>Dixie Chicks</td>
</tr>
<tr>
<td>Single Quote</td>
<td>\’</td>
<td>“Press \’y\’ for yes”</td>
<td>Press ‘y’ for yes</td>
</tr>
<tr>
<td>Double Quote</td>
<td>\”</td>
<td>“Homer said \”Doh\””</td>
<td>Homer said “Doh”</td>
</tr>
<tr>
<td>Backslash</td>
<td>\</td>
<td>“Here is a backslash, i.e., a \ character”</td>
<td>Here is a backslash, i.e., a \ character</td>
</tr>
</tbody>
</table>
null and Empty Strings

- If you just declare a String, as follows:
  ```javascript
  var s1: String;
  ```
  it is null (not referring to a value yet)
- The following String IS referring to a value:
  ```javascript
  var s2: String = "";
  ```
  It is an empty String (s2.length == 0).
- A String with a space in it is not empty:
  ```javascript
  var s3: String = " ";
  ```
  It is a String with one char (s3.length == 1).
Some String Properties and Methods

```typescript
var s1: String; var s2: String = “DoReMi”; var s3: String = “FaSo”;
```

<table>
<thead>
<tr>
<th>Method or Property Name</th>
<th>Purpose</th>
<th>Example</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>+</code></td>
<td>Join strings together (concatenation)</td>
<td><code>s1 = s2 + s3;</code></td>
<td><code>s1 = “DoReMiFaSo”</code></td>
</tr>
<tr>
<td><code>length</code></td>
<td>Find number of chars in a String</td>
<td><code>var len: int = s2.length;</code></td>
<td><code>len = 6</code></td>
</tr>
<tr>
<td><code>indexOf( s )</code></td>
<td>Find where substring s is located in a String</td>
<td><code>var pos: int = s3.indexOf(“So”);</code></td>
<td><code>pos = 2</code></td>
</tr>
<tr>
<td><code>substring(m,n)</code></td>
<td>Select the substring between positions m and n-1 in String</td>
<td><code>var sub: String = s2.substring(1, 5);</code></td>
<td><code>sub = “oReM”</code></td>
</tr>
</tbody>
</table>
### Some String Methods

```javascript
var s1: String;

var s2: String = "DoReMi";

var s3: String = "doremi";  // Note: The characters “D” and “d” are not the same
```

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Purpose</th>
<th>Example</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>charAt(n)</td>
<td>Returns the character at position n in a String</td>
<td>var ch: String = s3.charAt( 4 );</td>
<td>ch = ‘m’</td>
</tr>
<tr>
<td>toUpperCase( )</td>
<td>Converts a String to be all upper-case</td>
<td>var newSt: String = s2.toUpperCase( )</td>
<td>newSt = &quot;DOREMI&quot;</td>
</tr>
<tr>
<td>toLowerCase( )</td>
<td>Converts a String to be all lower-case</td>
<td>var newSt: String = s2.toLowerCase( );</td>
<td>newSt = &quot;doremi&quot;</td>
</tr>
</tbody>
</table>
The Shape class is used for drawing vector shapes: circles, lines, rectangles.

The Shape class has many properties, for instance:
- x and y position, which are ints
- graphics property, which is a Graphics object

After creating a Shape object, you use its graphics property to access methods of the Graphics class to draw a shape.

You can’t create a Graphics object directly. It’s only within other objects: Shape, MovieClip, …
## Methods of the Graphics Class

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>beginFill</td>
<td>Specifies fill color and alpha of a shape</td>
</tr>
<tr>
<td>clear</td>
<td>Clears graphics drawn in a Graphics object</td>
</tr>
<tr>
<td>drawCircle</td>
<td>Draws a circle with specified position &amp; radius</td>
</tr>
<tr>
<td>drawRect</td>
<td>Draws a rectangle with specified position, width &amp; height</td>
</tr>
<tr>
<td>drawRoundRect</td>
<td>Same, but also with corner ellipse width and height</td>
</tr>
<tr>
<td>endFill</td>
<td>Applies a fill to all shapes drawn since last call to beginFill</td>
</tr>
<tr>
<td>lineStyle</td>
<td>Specifies border thickness, color, and other options</td>
</tr>
<tr>
<td>lineTo</td>
<td>Draws line from current drawing position to specified position &amp; resets current drawing position</td>
</tr>
<tr>
<td>moveTo</td>
<td>Resets current drawing position to specified position</td>
</tr>
</tbody>
</table>
Objects in a class may themselves contain an object. For example, Shape objects contain a Graphics object called “graphics”, to which drawing commands can be applied.

Example:

```javascript
var square: Shape = new Shape();
square.graphics.drawRect(0,0,200,200);
square.x = 50;
square.y = 40;
```
Drawing a Square

```javascript
var square: Shape = new Shape( );
square.graphics.beginFill( 0x0000ff ); // blue
square.graphics.lineStyle( 4, 0xff0000 ); // red
// Draw a blue square with red border
square.graphics.drawRect( 10, 10, 200, 200 );
square.graphics.endFill( );
addChild( square ); // only in display list now
```
Drawing Lines

```javascript
var line: Shape = new Shape();
line.graphics.lineStyle( 6, 0x00ff00 ); // green
line.graphics.moveTo( 130, 140 ); // initial position
// Draw 3 connected green lines
line.graphics.lineTo( 230, 210 );
line.graphics.lineTo( 300, 150 );
line.graphics.lineTo( 30, 240 );
addChild( line );
```