CISC 110 Week 9

Extra Features in Flash and Test Review
Today

- Storyboards due
- Test 2 Solutions posted
- Game Math and Trigonometry: See Text!
- Custom Classes
- Adding and Removing Objects Dynamically
- Arrays of Objects
Game Math and Trigonometry

See Text:

- 3.7, pp. 139-142
- 5.8, Case Study 3, pp. 262-268
- 5.10, Case Study 4, pp. 273-281
  - Also uses variation of If statement: Switch
- 5.11, Case Study 5, pp. 282-291
Logon to Caslab!

- And follow along....
Custom Classes

When you create a new MovieClip symbol, for instance Orange, you are creating a custom class. To create an instance of the symbol, you drag the MovieClip on the stage. You can then give it an instance name, for instance orange1, to refer to it in ActionScript, via the Properties Panel.

Here’s how to create an instance of Orange called orange1 dynamically (i.e. via ActionScript):

```javascript
var orange1: Orange = new Orange();
```

To do this, there must be a linkage to an Orange class definition.
Custom Class Linkage: Method 1

1. Select Insert | New Symbol. In the dialog box, give it a name, e.g., Orange, and specify its type as MovieClip, select “Export for ActionScript” and add a name to the Class field, e.g., Orange.

2. When you click ok, it will tell you that it’s going to create a definition for the class. Click ok again.

3. Create graphics for your MovieClip as usual.

4. Now you can create an instance of your custom class:
   ```javascript
   var orange1: Orange = new Orange();
   ```
Custom Class Linkage: Method 2

1. Start with a symbol that you’ve created that’s in your library, e.g., a movie clip called Orange.

2. In the library panel, click the Symbol Properties button (looks like an “i” at the bottom of the library) or choose Properties from the library menu.

3. In resulting dialog, click to enable the Export for ActionScript option and add a name to the Class field, e.g., Orange.

4. Now you can create an instance of your custom class:
   var orange1: Orange = new Orange();
Adding Objects to the Display List Using the `addChild()` Method

**Purpose:** to add an object to the display list dynamically

```javascript
// Create an instance of a custom class
var orange1: Orange = new Orange();

// Add the object to the stage
addChild( orange1 );
```
Using the addChild() Method

Example: Add new orange to display list with each mouse click

```javascript
var inc: int = 0;
stage.addEventListener( MouseEvent.CLICK, onClick );

function onClick( evt: MouseEvent )
{
    var snack: Orange = new Orange();
    snack.x = snack.y = 100 + inc * 10;
    addChild( snack );
    inc ++;
}
```
Adding Objects to the Display List Using the addChildAt() Method

**Purpose:** to add an object at a specific depth (i.e. layer) in the display list

```javascript
// Create an instance of a custom class
var orange1: Orange = new Orange();

// Add object to stage at the lowest display depth
addChildAt( orange1, 0 );
```
Using the `addChildAt()` Method

Example: Add new orange at the lowest display depth with each mouse click

```javascript
var inc: int = 0;
stage.addEventListener(MouseEvent.CLICK, onClick);

function onClick( evt: MouseEvent ): void {
    var snack: Orange = new Orange();
    snack.x = snack.y = 100 + inc * 10;
    addChildAt( snack, 0 );
    inc ++;
}
```
Removing Objects from the Display List

To remove a specific object from the display list, you can use the `removeChild()` method.

Example: Remove the snack object

```javascript
removeChild( snack );
```

To remove a display object at a specific depth, you can use the `removeChildAt()` method.

Example: Remove the display object at depth 0

```javascript
removeChildAt( 0 );
```
Removing Objects from the Display List

**Example:** Add 20 oranges. Then remove the bottom one with each mouse click.

```javascript
for( var inc: int = 0; inc < 20; inc++ )
{
    var snack: Orange = new Orange();
    snack.x = snack.y = 100 + inc * 10;
    addChild( snack );
}

stage.addEventListener( MouseEvent.CLICK, onClick);
function onClick( evt: MouseEvent ): void
{
    removeChildAt( 0 );
}
```
Remove Objects from Memory Too!

Example: Add an orange. Then remove it with mouse click.

```javascript
var snack: Orange = new Orange();
snack.x = snack.y = 100;
addChild(snack);

stage.addEventListener(MouseEvent.CLICK, onClick);

function onClick(evt: MouseEvent): void {
    removeChild(snack); // Remove from display list (remove from stage)
    snack = null; // Remove from memory (delete instance)
}
```
Using Display Object Containers to Position Groups of Objects

```javascript
var orangeCrate: MovieClip = new MovieClip();
var orange1: Orange = new Orange();
var orange2: Orange = new Orange();
orange2.x = orange1.x + orange1.width + 10;

addChild( orangeCrate );
orangeCrate.addChild( orange1 );
orangeCrate.addChild( orange2 );

function onClick( evt: MouseEvent ): void
{
    orangeCrate.x += 10;
    orangeCrate.y += 10;
}
```
Managing Object Depths

// Assume we’ve already created a Pepper class
var fruit: Orange = new Orange();
var veggie: Pepper = new Pepper();
addChild( fruit );
addChild( veggie );

// Swap the depths of fruit & veggie
swapChildren( fruit, veggie );
trace( “fruit depth: ” + getChildIndex( fruit ) );
trace( “veggie depth: ” + getChildIndex( veggie ) );
Other Depth-Management Methods

`getChildAt( )`: Returns a DisplayObject instance at a specified index

`setChildIndex( )`: Changes the index position of the specified Child DisplayObject

`swapChildrenAt( )`: Swaps the positions of two child DisplayObjects at specified index positions
Arrays of Objects

// A declaration of an empty array
var list: Array = new Array();

// Add an object to the array and to the stage
var alien: AlienHead = new AlienHead();
alien.y = Math.random() * 300 + 100;
alien.x = 500;
list.push(alien);
addChild(alien);
Arrays of Objects

// Frame Event: move all aliens 5 pixels to left
stage.addEventListener(Event.ENTER_FRAME, moveAlien);

function moveAlien(evt: Event)
{
    for( var i = 0; i < list.length; i ++ )
    {
        list[i].x = list[i].x - 5;
    }
}
CISC 110 Test 3 Info

- **When:** Tuesday, November 17th, 6:30pm-
- **Where:** Here
- **What:** Everything covered in Test 1 and Test 2 + functions (functions with parameters and functions that return results)
  - Programming in ActionScript only, nothing on animation
  - Study custom notes, lecture slides, exercises, labs and assignments
- **You may bring 1 single sided cheat sheet, hand written not typed. Really.**
1 (long) Question with subparts:
- Ex. Write a program that does (insert specific task)
- Similar to assignments (but simpler)
- You should be able to finish in 1hr, but you have the full 3hr lecture slot
Question 1

Write a function called `reverse` with an Array parameter called `list` that returns a `new` array that contains the same elements as in `list`, except that the elements are in reverse order. The list may contain any number of elements. Do this `without` using the array method `reverse()`.
Question 2

Write a function called `findMe` that takes an array called `list` and a string called `name` as parameters and returns the index of `name` if it is in the array and -1 otherwise. Do this **without** using the array method `indexOf()`. 
Question 3

a) Create an array of strings called myNames and fill it with 5 names

b) Insert “Billy” into a random position in myNames

c) Using your functions from Questions 1 and 2, output the position of “Billy” in the output window. Reverse the order of myNames and output the new position of “Billy” in the output window