ACTIONSCRIPT SESSION 2

Roy McElmurry

Constants

Syntax:

private const **NAME**:**type** = **value**;

- Constants cannot be changed after they are initialized and must be initialized when declared

More: Constants

Casting

Syntax:

```
var num:Number = 23.5;
var newNum:int = num as int;
```

- There is also a more Java-like syntax

  (int)(num), but the former way is better for several subtle reasons

More: http://ragona.com/blog/as3-casting-vs-typevar/

Default Parameters

Syntax:

public function **name**(**param1**:**type** = **value**, …) {}

- If you make default a parameter, all parameters thereafter must also be defaulted
- The only way to give a value to the nth defaulted parameter is to give a value to all the preceding parameters as well

More: Default Parameters

Stage

- The stage is an object that stores all of the items that will be displayed on the screen
- When you run your program, your class is instantiated and added as the first child of the stage object
- The stage object can be referenced by any *DisplayObject* that is on the display list
- The stage contains some very useful fields that we can alter

More: http://www.adobe.com/livedocs/flash/9.0/ActionScriptLangRefV3/flash/display/Stage.html

Stage Fields

- Some of these fields are the same ones we are setting in the SWF command, but there are many others that we can set manually

Field	
stage.stageHeight	Height of the flash movie
stage.stageWidth	Width of the flash movie
stage.frameRate	Frame rate of the flash movie
stage.scaleMode	How the movie reacts when resized
stage.quality	The quality setting of the movie

More: http://www.adobe.com/livedocs/flash/9.0/ActionScriptLangRefV3/flash/display/Stage.html

DisplayObjectContainer

- An object that can hold things as children for displaying on the screen

- Examples: *Stage*, *Sprite*, *MovieClip*

More: http://www.adobe.com/livedocs/flash/9.0/ActionScriptLangRefV3/flash/display/DisplayObjectContainer.html

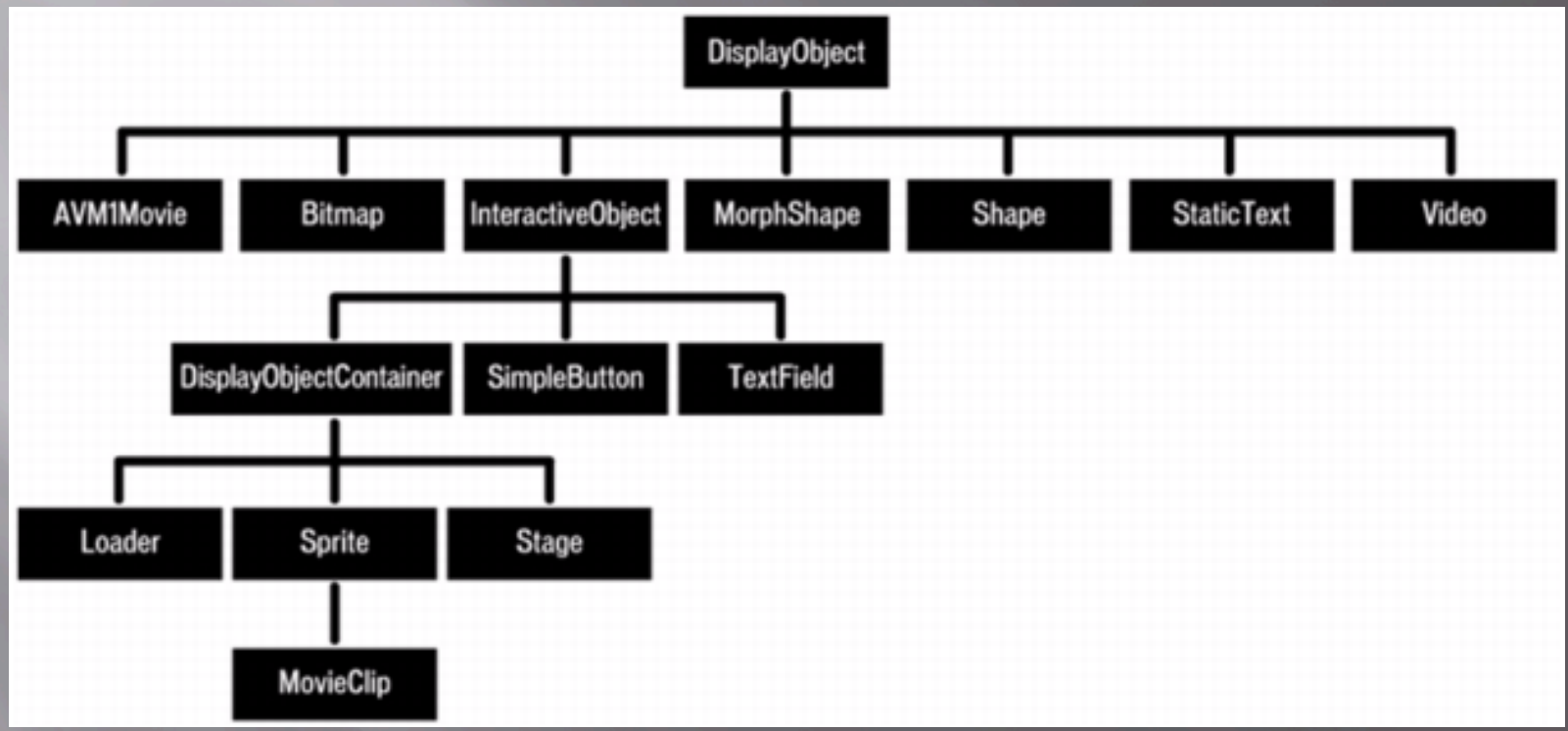
DisplayObject

- An object that can be displayed on the screen
- Anything inheriting from this can be drawn onto the screen if added to a *DisplayObjectContainer*

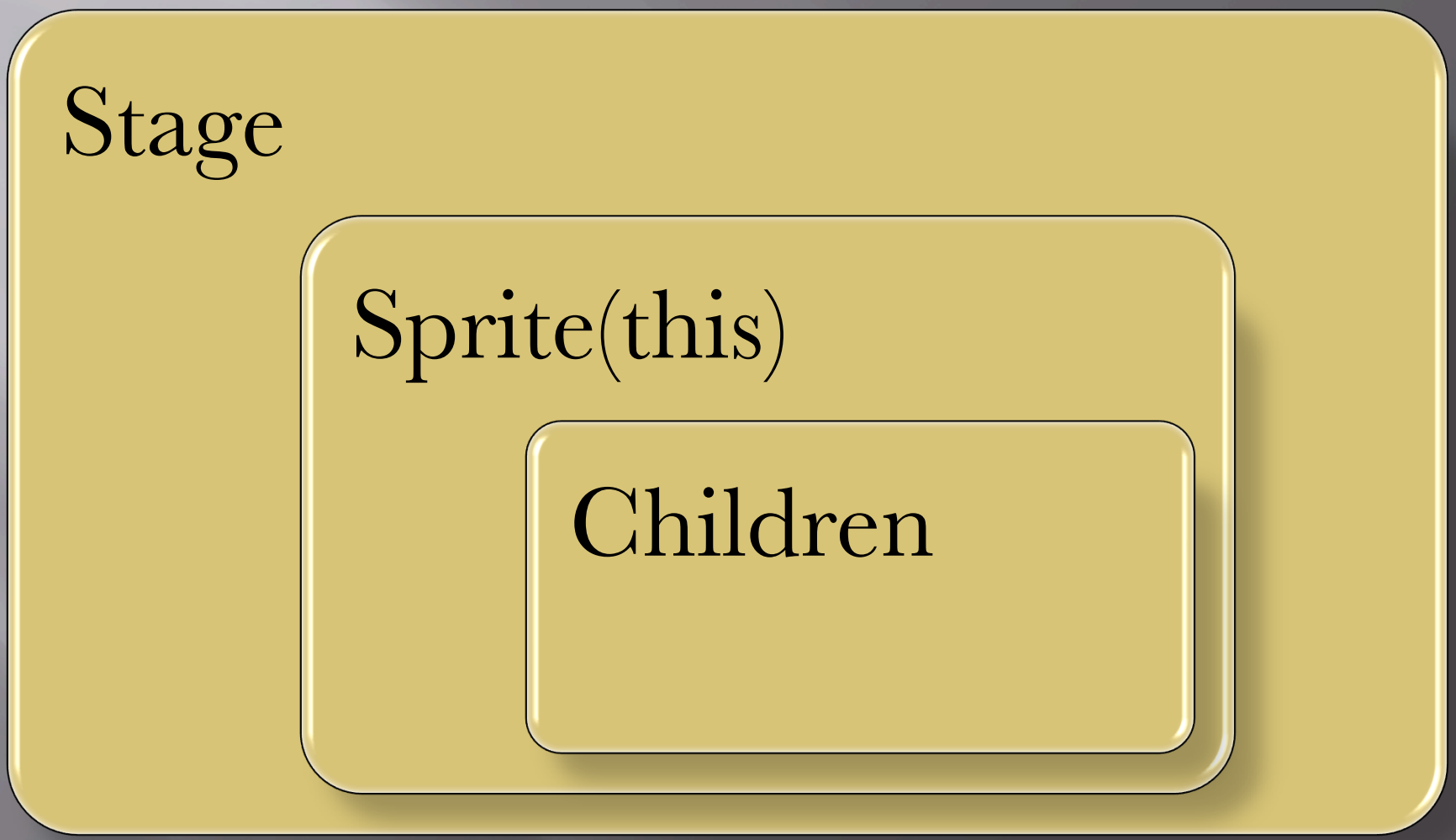
- Examples: *Shape*, *TextField*, *Sprite*

More: http://www.adobe.com/livedocs/flash/9.0/ActionScriptLangRefV3/flash/display/DisplayObject.html

Relationship



Display List



Display List

- The display list is simply a list of the things that will be displayed on the screen
- Using the *addChild*() is one way to add something to the display list
- When children are added to the same *DisplayObjectContainer*, they are automatically assigned depths starting at 0
- The order in which things are drawn on screen cooresponds to the depths of their parents
- An instance of your class is the first child of stage with a depth of 0

More: http://www.republicofcode.com/tutorials/flash/as3displaylist/

DisplayObjectContainer Functions

- These functions allow us to manipulate the order in which things are draw to the screen by manipulating the depth of children on *DisplayObjectContainer*'s

Functions	
cont.addChildAt(obj:DisplayObject, depth:int)	Add a child at the given depth between 0 and cont.numChildren
cont.setChildIndex(obj:DisplayObject, depth:int)	Set the depth of a child anywhere from 0 to cont.numChildren exclusive
cont.getChildIndex(obj:DisplayObject)	Returns the depth of the DisplayObject
cont.swapChildrenAt(depth1:int, depth2:int)	Swaps the children at depth1 and depth2
cont.removeChild(obj:DisplayObject)	Removes obj from cont and shifts deeper children down

Point

Syntax:

```
import flash.geom.Point;
…
var name:Point = new Point(x:Number, y:Number);
```

Functions	
Point.distance(p1:Point, p2:Point);	Returns the distance between the two points
Point.polar(len:Number, angle:Number);	Returns a Point with cartesian coordinates, where angle is in radians
p1.add(p2:Point);	Add the coordinates of p2 to p1
p1.offset(dx:Number, dy:Number);	Offset p1 by dx and dy

More: http://www.adobe.com/livedocs/flash/9.0/ActionScriptLangRefV3/flash/geom/Point.html

Timer

Syntax:

```
import flash.utils.Timer;
…
var name:Timer = new Timer(delay:Number, repeats:int = 0);
```

Functions	
timer.start();	Starts the timer
timer.stop();	Stops the timer
timer.reset()	Resets the timer
timer.addEventListener(type:String, listener:Function);	Registers timer for an event

More: http://www.adobe.com/livedocs/flash/9.0/ActionScriptLangRefV3/flash/utils/Timer.html

Events

- Events are flags that are raised by certain objects for different purposes
- In many cases we want something to happen when this event occurs, some such events may include
  - A button being clicked
  - The user pressing a key on the keyboard
  - A picture having finished loading
  - A timer going off
- When we use addEventListener we are saying that we wish for the listener function to be executed when the given type of event occurs

Timer events

Syntax:

```
import flash.events.TimerEvent.*;

timer.addEventListener(TimerEvent.TIMER, listener);
timer.addEventListener(TimerEvent.TIMER_COMPLETE, listener2);
```

- The *TimerEvent.TIMER* event happens when the timer reaches the time delay it was given
- The *TimerEvent.TIMER_COMPLETE* event happens when the timer has wrung the number of times specified

More: http://www.adobe.com/livedocs/flash/9.0/ActionScriptLangRefV3/flash/events/TimerEvent.html

Listener Functions

Syntax:

```
public function listener(ev:Type):void {
    //do stuff
}
```

- The name of the listener function must exactly match the function name given to *addEventListener()*
- The listener function must take a single parameter of the same type as that specified when you used *addEventListener()*
- The parameter contains valuable information about the event that one might want to use

Animation

- Events are essential to any interactive flash movies
- We can use a Timer to animate things rather easily by listening for the *TimerEvent.TIMER* event
- There are also events for mouse clicks and keyboard strokes that can used to create interactive games
- Many more events are also at your disposal

Loading an Image

Syntax:

```
import flash.display.Loader;
import flash.display.Bitmap;
…
var loader:Loader = new Loader();
loader.contentLoaderInfo.addEventListener(Event.COMPLETE, loaded);
loader.load("myimage.jpg");
…
public function loaded(e:Event):void {
        var img:Bitmap = new Bitmap(e.target.content.bitmapData);
        addChild(img);
}
```

Embed Resources

Syntax:

[Embed(source="**/filename.ext**")]
private var **name1**:Class;
private var **name2**:**Type** = new **name1**() as **Type**;

- Using the embed command is a shortcut for loading a resource and responding to it's load complete event
- This command must be placed in the fields area of your class
- In practice it would be better style to initialize **name2** in the constructor

Sound

Syntax:

```
import flash.media.Sound;
…
[Embed(source="/filename.mp3")]
private var name1:Class;
private var name2:Sound = new name1() as Sound;
```

Functions/Fields	
sound.play();	Plays the mp3, returns a SoundChannel object which could be used to tinker with the sound
sound.id3	Contains metadata about the sound such as artist, duration, title, etc.
sound.length	Returns the length of the mp3

More: http://www.adobe.com/livedocs/flash/9.0/ActionScriptLangRefV3/flash/media/Sound.html

Bitmap

Syntax:

```
import flash.display.Bitmap;
…
[Embed(source="/filename.jpg")]          //also png or gif
private var name1:Class;
private var name2:Bitmap = new name1() as Bitmap;
```

Functions	
pic.getBounds(obj:DisplayObject)	Returns a Rectangle object that defines the area of pic relative to obj
pic.hitTestObject(obj:DisplayObject);	Determines if pic is intersecting the given DisplayObject
pic.hitTestPoint(x:Number, y:Number);	Determines if pic intersects (x, y)

More: http://www.adobe.com/livedocs/flash/9.0/ActionScriptLangRefV3/flash/display/Bitmap.html

Collisions

- In games it is often important to know when objects on the screen are touching
- Even the simplest games need to respond to objects colliding with either other objects or specific points
- Examples
  - Pong – ball bounce
  - Worm – eating an apple
  - Galaga – bullets hitting enemies
  - Halo – pwning newbs