

Berne University of Applied Sciences

Flash - ActionScript 3

Dr. E. Benoist

Bibliography: <http://as3.betaruce.com/tut/>
<http://livedocs.adobe.com/flex/201/langref/index.html>

Fall Semester 07/08

Table of Contents

- Actionscript's principles
- Actionscript Syntax
 - Helloworld
 - Creating a Movie Clip instance
 - Event Handling
 - Import an image
- Action Script Syntax
- Flex: develop in AS without Flash IDE
- Bibliography

ActionScript's principles

▶ **Syntax: from Javascript to Java**

- Actionscript was just there for some small interaction (`stop()`, `gotoAndPlay(frame)`)
- It developed using Javascript syntax: loosely typed, eventhandling attached to the objects by clicking on them
- Now mature: more Object Oriented, less loosely typed
- Classes organized in packages, Javadoc like documentation, ...
- Programs are stored in separate files.

▶ **Symbols are classes derivating from basis classes**

- `MovieClip` is the central concept of any animation

▶ **A `MovieClip` class should be attached to the “Flash Animation”**

Helloworld Example

- ▶ **To create a class, we create an Action Script file with its name.**
 - class `Example` will be stored in file `Example.as`
- ▶ **We have to link a Class extending `MovieClip` with our main movie clip.**
 - We need to integrate the actionscript file in a `.fla` file to compile it into a `.swf` file.
 - The constructor is executed when the Movie is started.

HelloWorld Example

- ▶ We create the file `helloWorld.as`
- ▶ We write the definition of the class
- ▶ We create a new `.fla` file and define `helloWorld` as `DocumentClass`.

```
package{  
    import flash.display.MovieClip;  
  
    public class helloWorld extends MovieClip{  
        public function helloWorld(){  
            trace('Hello');  
        }  
    }  
}
```

Action Script Syntax

▶ Action Script Creates a Tree,like Document Object Model in HTML

- The root is the MovieClip Object attached to the application
- We can attache children to this object (= inserting an object on its stage).

▶ We can also create empty new instances of a MovieClip and configure them after the creation

- We use the graphics member of a MovieClip instance to draw in it.
- Graphics knows how to: `beginFill(color,alpha)` (starts the fill mode), `curveTo` (draw a curve from the current position to the given position), `drawCircle()`, `drawRect()`, `drawRoundRect()`, `endFill()` (go back to draw only mode)
- ...

Movie Clip including 2 MCs

```
package{
    import flash.display.MovieClip;
    public class example extends MovieClip{
        public var mc1:MovieClip = new MovieClip();
        public var mc2:MovieClip = new MovieClip();
        public function example(){
            mc1.graphics.lineStyle(1);
            mc1.graphics.beginFill(0xff0000);
            mc1.graphics.drawCircle(100,100,50);
            this.addChild(mc1);
            mc2.graphics.lineStyle(1);
            mc2.graphics.beginFill(0xffff00);
            mc2.graphics.drawRect(100,100,150,100);
            this.addChild(mc2);
        }
    }
```

Event Handling in AS3

- ▶ **We can add event handling to any instance of the class `MovieClip`.**
- ▶ **We call the method `addEventListener` on the object**
 - First argument: the event
 - Second argument: the function that has to be executed

```
mc1.addEventListener(Event.ENTER_FRAME, enterFrame_handler);
```

- ▶ **Available Events for a movie Clip**
 - `Event.ENTER_FRAME` fired as long as the clip is not stopped, by each frame entry.
 - `MouseEvent.CLICK`
 - `MouseEvent.DOUBLE_CLICK` (Only if activated)
 - `MouseEvent.MOUSE_OVER`
 - ...

Change properties of MC Instances

- ▶ **MovieClips are children of a parent**
- ▶ **They have properties:**
 - `x` and `y` are read write properties: place of MC in its parent stage (read/write).
 - `mouseX`, `mouseY` indicates where the mouse is currently;
 - `currentFrame` (read only) indicates at which frame this MC is.
 - `visible` (Read/Write) to change the visibility status of an object.
 - `width` and `height`(RW) of the object.

Example

```
package{
    import flash.display.MovieClip;
    public class example extends MovieClip{
        public var mc1:MovieClip = new MovieClip();
        public var mc2:MovieCLip = new MovieClip();

        public function example(){
            mc1.graphics.lineStyle(1);
            mc1.graphics.beginFill(0xff0000);
            mc1.graphics.drawCircle(100,100,50);
            mc1.addEventListener(Event.ENTER_FRAME,enterFrame_handler);
            this.addChild(mc1);
        }
    }
}
```

```
...
mc2.graphics.lineStyle(1);
mc2.graphics.beginFill(0xffff00);
mc2.graphics.drawRect(100,100,150,100);
mc2.addEventListener(MouseEvent.CLICK, mouseClicked_handler);
this.addChild(mc2);
}
private function enterFrame_handler(e:Event):void{
    mc1.x += 3;
}
private function mouseClicked_handler(e:Event):void{
    trace("mc2_Rectangle_is_clicked!");
}
}
```

Import an Image

- ▶ **We import the image in the flash application**
 - File/Import/ImporttoStage
 - select the image
 - Modify/ConverttoSymbol(F8)
 - Select type Movie-Clip
 - Remove the image from the stage
- ▶ **Create a class representing your MC**
 - Open the library (Ctrl L)
 - Right click on your image, choose Linkage
 - Choose Export for ActionScript
 - Type the name of the class you want to define
- ▶ **We can create an instance of an existing MC Class**

```
mc = new MyImageClass();
```

More on Event Listeners

- ▶ **We can add an event listener**

```
mc.addEventListener(MouseEvent.DOUBLE_CLICK,  
doubleClick_handler);
```

- ▶ **We can also remove an event listener**

```
mc.removeEventListener(MouseEvent.DOUBLE_CLICK,  
doubleClick_handler);
```

- ▶ **The event listener function can access the Object which generated this event: `event.target`**

Example

- ▶ We already have imported an image, created a MC symbol and named its class FunnyCar

```
package{
import flash.display.*;
import flash.events.*;
public class Car extends MovieClip{
    public var mc:MovieClip;
    public function Car(){
        mc = new FunnyCar();
        this.addChild(mc);
        mc.y = stage.stageHeight/2;
        mc.doubleClickEnabled = true;
        mc.addEventListener("doubleClick", doubleClick_handler);
        mc.addEventListener("click", click_handler);
    }
}
```

```
...
private function doubleClick_handler(e:Event){
    e.target.addEventListener("enterFrame",enterFrame_handler);
}
private function enterFrame_handler(e:Event){
    e.target.x += 3;
}
private function click_handler(e:Event){
    e.target.removeEventListener("enterFrame", enterFrame_handler);
}
}
}
```

Use text Field ¹

- ▶ We can create an instance of the `flash.text.TextField` class
- ▶ It has a member `text` that is read/write

```
public function HelloWorld(){  
    var display_txt:TextField = new TextField();  
    display_txt.text = "Hello World!";  
    addChild(display_txt);  
}
```

¹this example comes from <http://www.senocular.com/flash/tutorials/as3withxmlc/>

Actionscript syntax

► Data Typing

- Type is given after the element `variable:Type`
- All members should be typed
- If a member can accept any type, use `*`

```
var myNum:Number;  
var myVar:*
```

- Return value should be typed, use `void` if the function returns nothing

```
function myFunction():void {  
}
```

Actionscript syntax (Cont.)

- ▶ **Parameters/Arguments: Functions can have optional value.**

```
public function testFunctions():void {
    usingOptional(1);
    // usingOptional(); <- wrong - first parameter required
    usingRest(1, 2, 3, 4);
}
private function usingOptional(required:Number,
                                optional:String = "default"):void {
    trace(required); // 1
    trace(optional); // "default"
}
private function usingRest(required:Number, ... optionalArgs):void {
    trace(required); // 1
    trace(optionalArgs); // [2, 3, 4]
```

Dynamic Objects with dynamic properties

- ▶ **Unlike in Java, AS objects may receive properties on the run**

```
// Create a dynamic object with dynamic property  
var obj:Object = new Object();  
obj.prop = "value";  
// delete dynamic property on obj using delete  
delete obj.prop  
// cannot delete obj, only able to set to null  
obj = null;  
// int, uint, Number and Boolean types cannot be deleted  
var intNum:int = 0;  
var uintNum:uint = 0;  
var numNum:Number = NaN;
```

Flex SDK

- ▶ **Compile AS without using Flash**
 - Need the Flex Standard Development Kit (SDK)
- ▶ **Available on all platforms**
 - Windows and Mac
 - Beta Version for Linux (plug-in for Eclipse available)
- ▶ **Swing like development**
 - Define a tree of movie clips
 - Difference: the time lines (one per mc)

Flex Builder

- ▶ **Compiler** `mxmlc.exe`
- ▶ **Works like** `javac`

```
mxmlc.exe -o output.swf "D:\MyDirectory\samples\MyClass.as"
```

Hello World Example

- ▶ The root element extends the class Sprite

```
package {  
    import flash.display.Sprite;  
    import flash.text.TextField;  
    public class HelloWorld extends Sprite {  
        public function HelloWorld() {  
            var display_txt:TextField = new TextField();  
            display_txt.text = "Hello World!";  
            addChild(display_txt);  
        }  
    }  
}
```

Graphics and drawing

```
package {  
    import flash.display.Sprite;  
    import flash.display.Shape;  
    import flash.display.Graphics;  
    public class Rectangles extends Sprite {  
        public function Rectangles() {  
            drawColoredRectIn(graphics, 0xFF0000);  
            var rect:Shape = new Shape();  
            drawColoredRectIn(rect.graphics, 0xFFFF00);  
            rect.x = 50;  
            rect.y = 50;  
            addChild(rect);  
        }  
    }  
}
```

```
private function drawColoredRectIn(target:Graphics, color:int):void {  
    target.lineStyle(1, 0x000000);  
    target.beginFill(color);  
    target.drawRect(0, 0, 100, 100);  
}  
}  
}
```

Importing Assets from Flash (without Flash IDE)

```
package {  
    import flash.display.Sprite;  
    public class EmbedAssets extends Sprite {  
        [Embed(source="images/trophy.png")]  
        private var TrophyImage:Class;  
        [Embed(source="swfs/satdish.swf")]  
        private var SatelliteAnimation:Class;  
        public function EmbedAssets() {  
            addChild(new SatelliteAnimation());  
            addChild(new TrophyImage());  
        }  
    }  
}
```

More about Flash and Flex

- ▶ **Flash language reference** (Javadoc like Description of standard packages)

http:

[//livedocs.adobe.com/flex/201/langref/index.html](http://livedocs.adobe.com/flex/201/langref/index.html)

- ▶ **Flash tutorials (Flash with AS3 / Flex)**

<http://www.senocular.com/flash/tutorials.php>

- ▶ **An easy tutorial to start with Actionscript**

<http://as3.betaruce.com/tut/>

- ▶ **A lot of tutorials showing some nice features of AS (in french ;))**

<http://www.zoneflash.net/tutoriaux.php>