ActionScript 3 – A Concise Introduction

About Flash and its eco-system

Flash, Flex, Air?

Flash is widely used on the internet. Today the majority of desktop-users have the Flash-plugin installed, most of which have the latest stable version. Flex and AIR are technologies that are built on top of Flash.

ActionScript?

ActionScript is a language based on ECMA-Script. Therefore it is pretty similar to JavaScript. The current version is ActionScript 3 (used since Flash CS3 and Flex 2). ActionScript 3 also borrows from other languages like Java.

Flash & Sourcecode

Where should(n't) I put my code? (Flash Professional Edition)

One approach is to put code on keyframes on your Flash IDE timeline. Another possibility is to create ActionScript-files (simple text files that end in ".as"). We'll mostly use the latter, as it has many advantages over timeline-scripts.

Timeline-Scripts

As you know, scripts can be put into keyframes on any layer in any movieclip of your project. That it *can* be put there does not mean you *should* put it there. The usual place for ActionScript is a layer of its own (usually the topmost layer). Furthermore it is common practice to place only simple code[1] into movieclips in your library. The lion's share of code should be on the stage's timeline, topmost layer. There's a simple reason for that: It's easy to lose track of all the independent code-snippets if you scatter them all over your project.

Script-Files

There's the possibility to put code into external files (ending in .as usually). These files can be included via the import-command. This is the very best way to write code, as it permits Version Control and simpler editing from the outside. The following line will include a file called com/example/Bubble.as (relative to your .fla-file).

```
import com.example.Bubble;
```

Imports belong at the top of your code. Most advanced editors will manage your imports automatically.

1 Commands like stop(); or gotoAndPlay(); can be put in any movieclip.

Code-Basics

Comments

When writing larger programs, you will need to include some comments in so that you or another programmer can understand your document. Comments for a single line start with two slashes „//“, comments for many lines start with “/*” and end with “*/”.

Comments can help you debug your program. If you are not sure, what causes the error try putting a comment-symbol in front of the line(s) in question to make it a comment (therefore stopping Flash from executing that code) if the error is gone, you may have found it.

Some of the explanations on this sheet will be in form of comments from now on.

```
// This comment is only for one line of code. It automatically
// ends at the end of the line so you need the slashes in every line
// You can even put a comment behind a „normal“ line of code like this:

trace("Hello world");   // This is my hello-world comment.

/* This comment can contain multiple lines of code. It is especially useful if  you encounter
an error in your program. Put the comment-symbols around it, and it will not be executed. That
way you can pinpoint the error easily. */
```

Trace

Trace may be the most useful command there is. Every Language has its own way of generating simple text output – trace() is flash's way to do this. You can put text into the brackets and it will show in the “output”-window. This is very useful when you're monitoring what your program does (we'll get to that later). It's a simple but powerful debugging technique to just trace a variable or some text.

```
trace("Hello world");
```

You may ask “why is tracing so cool?” Well – tracing is only the first step. You can build a skeleton-application fast and then build the graphics and sounds and animations around it. Remember: if there's a working trace() in place, you may as well replace it with something else later!

Variables

Think of variables as place-holders, envelopes or boxes. You can use them to store something you'll need later.

```
// let's put something into this variable
var text:String = "a is smaller.";
trace(text)
// this will have outputted “a is smaller”
```

This example hardly shows the immense power behind variables. You could just as easily have traced the text directly. Let's skip ahead for the next example. We'll read the content of a textbox called `password`. The following code-snippet won't work on it's own.

```
var text:String = password.text;
trace(text);
// this will have traced the contents of a password-textbox
```

Variables have a scope[2], which means that you can't access all variables everywhere. Likewise it means

2 http://help.adobe.com/en_US/ActionScript/3.0_ProgrammingAS3/WS5b3ccc516d4fbf351e63e3d118a9b90204-7f9d.html

that you are free to reuse variablenames. The concept of scope is beyond the scope of this handout.

A simple decision

If the fridge is empty, I'll order pizza. Simple isn't it? Your code can do this, too.

```
// the following lines will only trace "I'm having what's in the fridge" the
// else-part is skipped
var itemsInTheFridge:Number = 2;

if (itemsInTheFridge > 0) {
    trace("Yummy, leftovers!");
} else {
    trace("It's empty, let's order pizza.");
}
```

Loops

Loops can perform repetitive tasks for you. Here's an example with while() and trace():

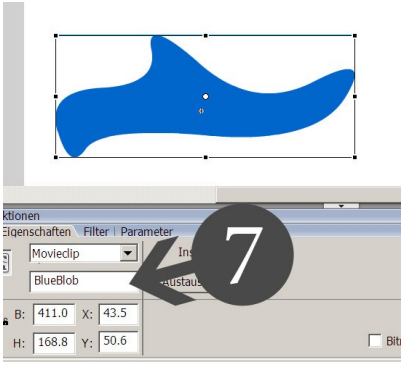
```
// the following lines will output „Hey, this round, i is ..." ten times.
var i:Number = 0;
while (i < 10) {
    trace("Hey, this round, i is " + String(i));
    i = i + 1;
}
```

And this is the same example with a for()-loop:

```
for(var i:Number = 0; i<10; i = i + 1) {
    trace("Hey, this round, i is " + i);
}
```

A movieclip's properties

All these examples assume, you have a movieclip with the instance name blueBlob. Select your object and look for the field marked with number 7 in the image to your right.




```
blueBlob.x = 100;        // X-Position in pixels
blueBlob.y = 120;        // Y-Position in pixels
blueBlob.height = 10;    // setting the height
blueBlob.width = 10;     // setting the width
blueBlob.rotation = 10; // rotation in degrees
                         // larger number -> clockwise
```

When you're altering these properties, you'll see the position, rotation or size of your movieclip change. A movieclip has many other properties which can be found in the AS3-Livedocs[3] from Adobe.

Functions

You're already using functions. If you have ever written a trace(), you've successfully used a function. That one, however has been provided by Flash itself. Let's see how we can define our own functions.

These functions may be as complex as you want. This is a very simple example for the sake of the short handout. You could easily make improvements like formatting of the numbers in your function. Functions come in handy, whenever you need to run the same code from many places.

3 http://livedocs.adobe.com/flash/9.0/ActionScriptLangRefV3/flash/display/MovieClip.html

```actionscript
// how many dollars will you get for a given value in euros?
function euroToDollar(euro:Number):String {
    return String(euro * 1.46) + " US$";
}
trace("10 Euros are " + euroToDollar(10));
```

Events

When you are clicking a button in Flash, it triggers an event. That means: The moment you press the button, Flash will run functions that have been registered as event-listeners. Events are used for a long list including mouse-, keyboard-, timer- and timeline-events.

```actionscript
// These are the functions we are going to use in the example.
// you can give them any name you like, but it's best to use expressive names like these
function keyDownHandler(event:KeyboardEvent):void {
    trace("down", event.keyCode);
}

function keyUpHandler(event:KeyboardEvent):void {
    trace("up", event.keyCode);
}

// Actually tying the knots together
stage.addEventListener(KeyboardEvent.KEY_DOWN, keyDownHandler);
stage.addEventListener(KeyboardEvent.KEY_UP, keyUpHandler);
```

Useful Events are: `Event.ENTER_FRAME` (code is run every frame), `MouseEvent.MOUSE_DOWN` (a mouseclick anywhere), `MouseEvent.MOUSE_UP` (release of the mousebutton), `MouseEvent.MOUSE_OVER` (hovering the mouse over this movieclip), `MouseEvent.MOUSE_OUT` (moving the mouse out of this movieclip's area).

Useful resources

http://actionscriptcheatsheet.com (cheat sheets)
http://flashforum.de (large german forum)
http://www.flashdevelop.org (free Flash-IDE)
http://www.adobe.com/devnet/edu.html (Free Flash-Builder for students, lots of resources)
http://www.adobe.com/devnet/actionscript.html (Actionscript 2 and 3 resources)

About this document

Version History

2007: Initial version of this document has been written specifically for AS2

2011-04-27: Major rewrite for AS3, keeping only the basic structure

The Latest version lives at http://docs.claudiuscoenen.de/programming/AS3-Concise-Introduction.odt

Contributors

Claudius Coenen, Christine Emrich, Oliver Göck, Garrit Schaap