

Daniel Bertram
Kerstin Götze

ASS Flex Workshop für die FH Köln, Campus
Gummersbach

16. Juni 2010

Syntax und Shortcuts

Shortcuts:

Alt + Pfeil hoch Pfeil runter	Zeile verschieben
Ctrl + Leertaste	Kodevervollständigung
Ctrl + .	Wort Vervollständigung
Tab	Einrücken
Shift + Tab	Einrückung reduzieren
cmd + D	Löschen der aktuellen Zeile
Shift + cmd + D	Erstellt einen CDATA Block

Syntax:

Klassen in MXML

```
<HBox  
    xmlns:mx="http://www.adobe.com/2006/mxml"  
    height="100"  
/>  
</HBox>
```

```
<Elternklasse  
    namespace:namespaceName="Pfad des Namespaces"  
    weiteresAttribut="Wert"  
/>  
</Elternklasse>
```

Klassen in AS3

```
package flex.workshop {  
    public class MyClass {}  
}
```

Schlüsselwort paketpfad {
 Sichtbarkeit Schlüsselwort KlassenName {}
}

Methoden in AS3

```
private function myMethod (parameter:String):Boolean {return true;}  
Sichtbarkeit Schlüsselwort methodenName (parameter:Datentyp):DatentypDesRückgabewertes
```

Variablen in AS3

```
protected var myVariable:Number = 0;  
Sichtbarkeit Schlüsselwort variablenName:Datentyp = Wert;
```

Kommentare in MXML

```
<!-- Ein Kommentar -->
```

Kommentare in AS3

```
// Ein einzeiliger Kommentar  
/* Ein  
mehrzeiliger  
Kommentar */
```

Daniel Bertram
Kerstin Götze

ASS Flex Workshop für die FH Köln, Campus
Gummersbach

16. Juni 2010

Syntax und Schlüsselwörter

DEFINITIONSSCHLÜSSELWÖRTER

var	Deklariert eine Variable <pre>var myVar:String;</pre>
const	Deklariert eine Konstante, also eine Variable, der nur ein einziges Mal ein Wert zugewiesen werden kann <pre>const myConst:int = 1;</pre>
package	Definiert ein Paket, in dem die Klasse organisiert ist. Dient der Strukturierung der Klassen. <pre>package myPackage { public class MyClass {} }</pre>
class	Definiert eine Klasse von der Instanzen erstellt werden können, die die von der Klasse definierten Methoden und Attribute besitzen. <pre>public class MyClass {}</pre>
interface	Definiert ein Interface. <pre>public interface MyInterface {}</pre>
function	Deklariert eine Methode. <pre>public function myFunction {}:String;</pre>
extends	Definiert, dass eine Klasse von einer anderen Klasse ableitet/erbt. <pre>public class MyClass extends HBox {}</pre>
implements	Definiert, dass eine Klasse ein oder mehrere Interfaces implementiert. <pre>public class MyClass implements MyFirstInterface, MySecondInterface {}</pre>
get	Deklariert einen Getter, d.h. eine Methode, die wie ein Attribut gelesen werden kann. <pre>public function get myVariable():String { return _myVariable; }</pre>
set	Deklariert einen Setter, d.h. eine Methode, die wie ein Attribut geschrieben werden kann.

```
public function set myVariable(value:String):void {
    _myVariable = value;
}
```

ATTRIBUTE

public	<p>Legt fest, dass eine Klasse, Methode, Konstante oder Variable für alle Aufrufer verfügbar ist.</p> <pre>public class MyClass {</pre>
protected	<p>Legt fest, dass eine Methode, Konstante oder Variable nur für die Instanzen der sie definierenden Klasse oder von dieser abgeleitete Klassen verfügbar ist</p> <pre>protected function myFunction():void {}</pre>
private	<p>Legt fest, dass eine Methode, Konstante oder Variable nur für die Instanzen der sie definierenden Klasse verfügbar ist</p> <pre>private var myVar:String;</pre>
static	<p>Legt fest, dass eine Methode, Konstante oder Variable zu einer Klasse gehört, statt zu den Instanzen der Klasse</p> <pre>public class MyClass { public static function myFunction():void {} }</pre> <p><u>Usage:</u> MyClass.myFunction();</p>
final	<p>Legt fest, dass eine Klasse nicht abgeleitet oder eine Methode nicht überschrieben werden kann.</p> <pre>final class MyClass {}</pre>
dynamic	<p>Legt fest, dass den Instanzen einer Klasse zur Laufzeit Attribute hinzugefügt werden können.</p> <pre>dynamic class RuntimeExtendableClass() {}</pre> <p><u>Usage:</u></p> <pre>var extClass:RuntimeExtendableClass = new RuntimeExtendableClass(); extClass.newProperty = "new";</pre>
override	<p>Legt fest, dass eine Methode, die gleichnamige von der Elternklasse geerbte Methode ersetzt.</p> <pre>override public function myFunction():void {}</pre>

SPEZIELLE DATENTYPEN UND KONSTANTEN

void	Legt fest, dass eine Methode keinen Wert zurückgibt <pre>public function myFunction():void {}</pre>
null	Ein spezielle Wert, der Variablen zugewiesen oder von Methoden zurückgegeben werden kann, wenn keine Daten vorhanden sind <pre>var value:String = null;</pre>
this	Eine Referenz in einer Klasse auf die konkrete Instanz dieser Klasse <pre>public class MyClass { private var property1:String; public static function myFunction():void { this.property1 = "test"; } }</pre>
NaN	Kann Variablen vom Typ Number zugewiesen werden, und sagt aus, dass der aktuelle Wert der Variablen "not a number", also keine Zahl ist. Dies kann zum Beispiel vorkommen, wenn einer Variablen vom Typ Number das Ergebnis einer Division durch Null zugewiesen wird. <pre>var result:Number = 100/0; trace (result); //Ausgabe: NaN</pre>

OPERATOREN

new	Erzeugt eine neue Instanz einer Klasse <pre>var instance:MyClass = new MyClass();</pre>
is	Prüft, ob eine Objekt von einem bestimmten Datentyp ist bzw. eine Instanz einer bestimmten Klasse oder eines Interfaces ist. Gibt auch dann true zurück, wenn das zu testende Objekt in seiner nicht direkt vom gesuchten Typ ist, jedoch direkt oder indirekt davon ableitet. <pre>var box:Hbox = new HBox(); trace(box is HBox); //Ausgabe: true trace(box is DisplayObject); //Ausgabe: true trace(box is VBox); //Ausgabe: false</pre>

DIREKTIVEN

import	Macht in anderen Paketen definierte Klassen in einer Klasse verfügbar <pre>package myPackage { import mx.controls.Button; }</pre>
include	Fügt den Inhalt einer externen Datei in den Code ein. <pre>include "CodeSnippet.as"</pre>

STATEMENTS

super	<p>Führt den Konstruktor der Elternklasse oder eine Methode der Elternklasse aus. Bei Methoden ist dieser Aufruf nur nötig, wenn diese Methode in der aktuellen Klasse überschrieben wurde, jedoch explizit die Methode aus der Elternklasse aufgerufen werden soll.</p> <pre>// Aufruf des Elternkonstruktors public function MyClass():void{ super(); } // Aufruf der Elternmethode super.myMethod();</pre>
for each ... in	<p>Iteriert durch alle Elemente eines Arrays</p> <pre>var myArray:Array = {"Hello", "Flex."}; for each (var item:String in myArray) { trace(item); } /* <u>Ausgabe:</u> Hello Flex. */</pre>
return	<p>Beendet die Ausführung eine Methode und kann einen Wert aus der ausführenden Methode zurückgeben.</p> <pre>private var myFunction_1():void{ return; } private var myFunction_2():int{ return 0; }</pre>

METADATEN

[Bindable]	Sorgt dafür, dass alle Methoden und Variablen einer Klasse bzw. eine einzelne Methode oder Variable als Quelle für ein DataBinding verwendet werden kann <pre>[Bindable] public var myProperty:int;</pre>
[Embed]	Bettet ein Bild, SWF, Sound,... ein <pre>[Embed(source="logo.png")] public var imgCls:Class;</pre>
[Event]	Definiert, dass eine Klasse ein bestimmtes Event wirft bzw. werfen kann. Dieses Tag ist nicht nötig, um Events zu werfen und per AS3 abzufangen. Soll das Event allerdings auch bei Verwendung der Klasse in einem MXML-Block direkt mit einem Handler belegt werden können, muss das Tag gesetzt sein. <pre>[Event(name="myClickEvent", type="flash.events.Event")]</pre>
[Style]	Definiert, dass eine Klasse ein bestimmtes Style Attribut besitzt und sorgt dafür, dass dieser Style auch bei Verwendung der Klasse in einem MXML-Block gesetzt werden kann. <pre>[Style(name="color", type="uint", inherit="yes")]</pre>
[ArrayElementType]	Definiert, dass ein Array nur Objekte eines bestimmten Typs aufnehmen darf. Grundsätzlich sind Arrays ungetypt und können gleichzeitig Objekte verschiedener Typen aufnehmen. <pre>[ArrayElementType("String")] public var myArray:Array;</pre>

KOMMENTARSYNTAX

//...	Kommentiert eine Zeile aus <pre>// einzeliger Kommentar</pre>
/*...*/	Kommentiert beliebig viele Zeilen aus <pre>/* mehrzeiliger Kommentar */</pre>

Globale Funktionen

trace()	Gibt eine Nachricht in der Konsole aus <pre>trace("test"); //Ausgabe: test</pre>
----------------	---

CASTS

Class(instanceToCast)	Cast von einem Objekt in einen bestimmten Datentyp. Wirft eine Fehler, wenn das Objekt nicht in diesen Typ umgewandelt werden kann.
instanceToCast as Class	Cast von einem Objekt in einen bestimmten Datentyp. Gibt null zurück, wenn das Objekt nicht in diesen Typ umgewandelt werden kann.

Daniel Bertram
Kerstin Götze

ASS Flex Workshop für die FH Köln, Campus
Gummersbach

16. Juni 2010

Tag 1 - Übung 2 - Online Doku

Aufgabenstellung:

Findet mit Hilfe der Online Dokumentation Alternativen zur Verwendung der Label Komponente und fügt sie der Anwendung hinzu.

Zusatzaufgabe(n):

- Fügt der Anwendung fünf weitere Komponenten eurer Wahl hinzu.
- Sucht in der Dokumentation nach einer Möglichkeit das TitleWindow per Mouse zu bewegen (Tip: nur ein spezielles Attribut muss gesetzt werden)

Hinweise:

- Verwendet nicht die Design View.
- Nehmt das Paket mx.controls unter die Lupe.

Code Snippets:

Keine.

Daniel Bertram Kerstin Götze	ASS Flex Workshop für die FH Köln, Campus Gummersbach 16. Juni 2010
---------------------------------	---

Tag 1 - Übung 3 - MXML & AS3

Aufgabenstellung:

Es liegen zwei Layoutskizzen vor. Anhand dieser sollen die dargestellten Komponenten umgesetzt werden.

Das Kalenderfenster sollte in MXML oder mit Hilfe des Design Views erstellt werden.

Das Konfigurationsfenster soll mit AS3 in einem Skriptblock erstellt werden.

Zusatzaufgabe(n):

- Am Schriftgrößen-Slider sollen in 8er-Schritten Skalenbeschriftungen angezeigt werden.
- Tage, die in der Vergangenheit liegen, dürfen im DateChooser nicht auswählbar sein.

Hinweise:

Keine.

Code Snippets:

// Einen Script Block erstellt ihr so:

```
<mx:Script>
    <![CDATA[
        import paket.KlassenName;
        private function nameDerMethode():void
        {}
    ]]>
</mx:Script>
```

// Um eine Methode direkt mit dem Start der Anwendung aufzurufen könnt ihr das „creation Complete“ Event benutzen:

```
<mx:Application creationComplete="init()">
<mx:Script>
    <![CDATA[
        // Und die Funktion die dann ausgeführt wird.
        private function init():void
        {
            Alert.show("application created");
        }
    ]]>
</mx:Script>
```

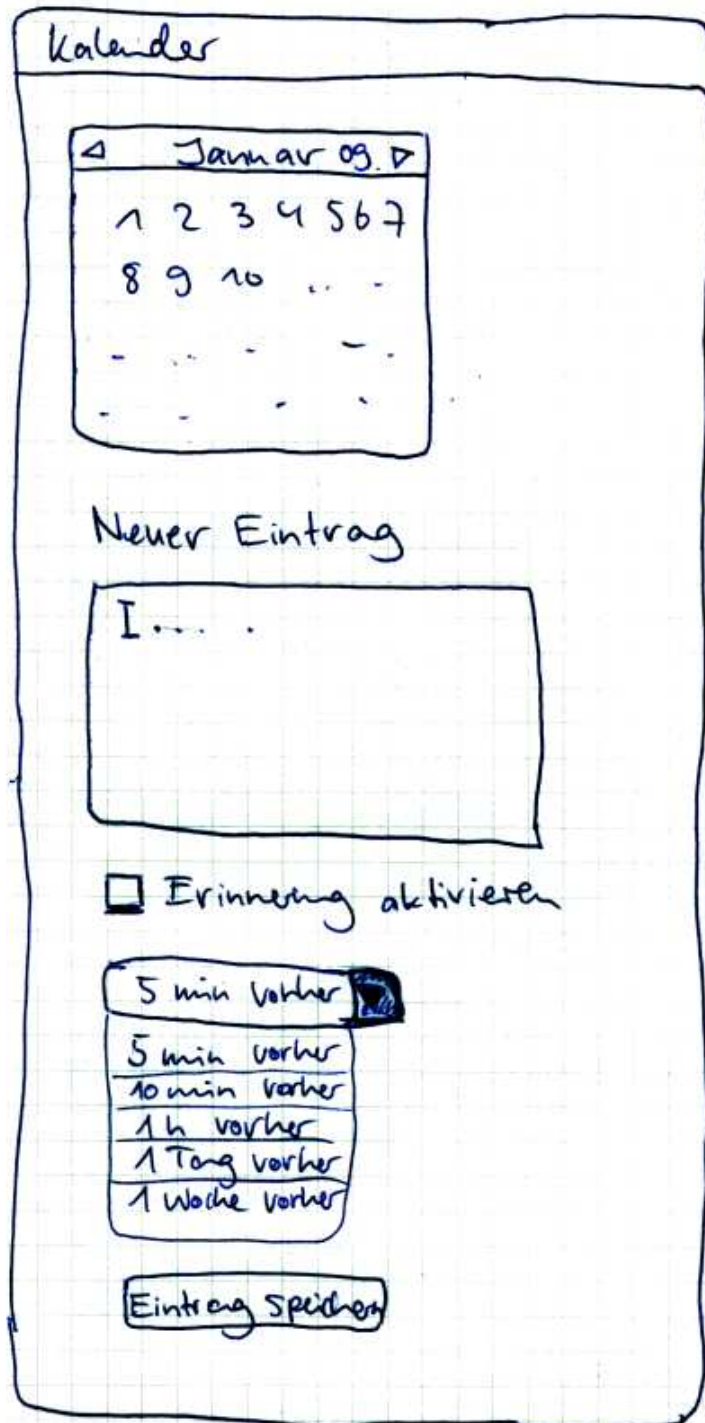
// Comboboxeinträge können mit folgender folgender Syntax erstellt werden

```
comboBox.dataProvider = ['Eintrag 1', 'Eintrag 2', '...'];
```

// Um ein GUI Objekt per AS3 einem Container hinzuzufügen geht so vor:

```
var guiObject:Label = new Label();
container.addChild(guiObject);
```


Screenshot Tab 1:



Screenshot Tab 2:

Kalenderkonfiguration


Farbschema



Akustische Benachrichtigung

Sound abspielen bei Erinnerung

Schriftgröße



Werte:
• (8 - 64)
• 16 vorangewählt

Daniel Bertram
Kerstin Götze

ASS Flex Workshop für die FH Köln, Campus
Gummersbach

16. Juni 2010

Tag 1 - Übung 4 - Layout Container

Aufgabenstellung:

Erstellt ein neues Projekt „FlexMediaPlayer“.

Die Applikation soll ein Fenster mit drei Tabs mit den Titeln „3D Properties“ und „Rotationseffekt“ enthalten. Das Fenster selbst hat den Titel Control Panel.

Auf den Aufgabenblättern seht ihr drei Screenshots, die den Inhalt der Tabs und dessen Anordnung zeigen. Baut diese Screenshots nach und nutzt die gerade vorgestellten Layout Container, um die dargestellte Anordnung der Controls zu erreichen.

Setzt bei allen Slidern **liveDragging** auf **true**.

Zusatzaufgabe(n):

- Erzeugt einen vierten Tab in dem sich ein Accordion befindet. Das Accordion hat drei Teile. In jedem Accordionteil befindet sich je ein Text, der es beschriftet.
- Erzeugt einen fünften Tab in dem sich in einer 3x3 Matrix angeordnet die folgenden Zeichen befinden: X, X, O, X, O, X, X, O, O. Verwendet dazu die Komponente Grid.

Hinweise:

- Informiert euch in der Online-Dokumentation über die Klasse `RadioButtonGroup`
- An jeder beliebigen Stelle im Code kann über `Application.application` auf die Hauptapplikation zugegriffen werden.

Code Snippets:

```
// Zugriff auf die Applikation
```

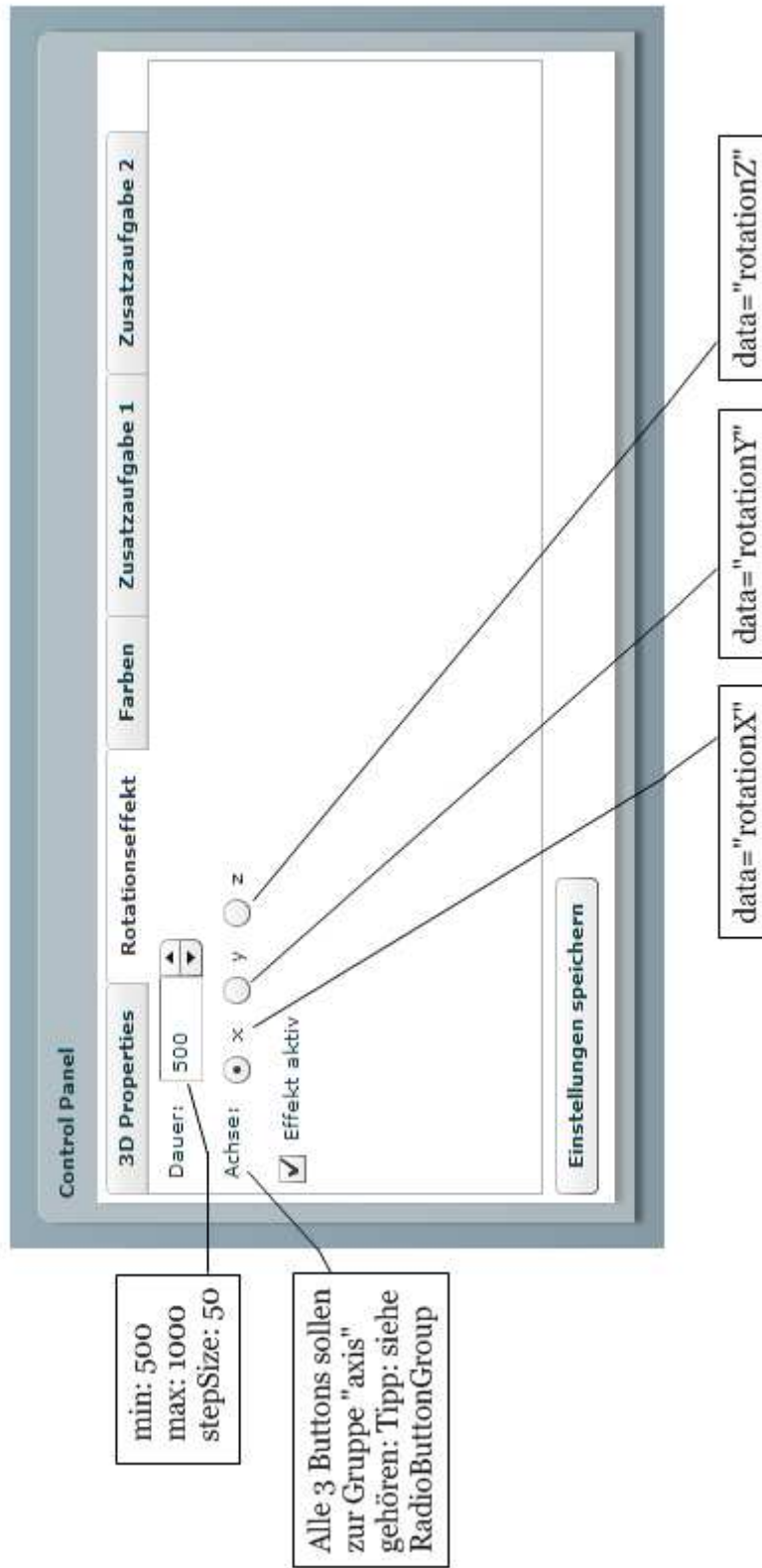
```
var app:Application = Application.application;
```

Screenshot 1:

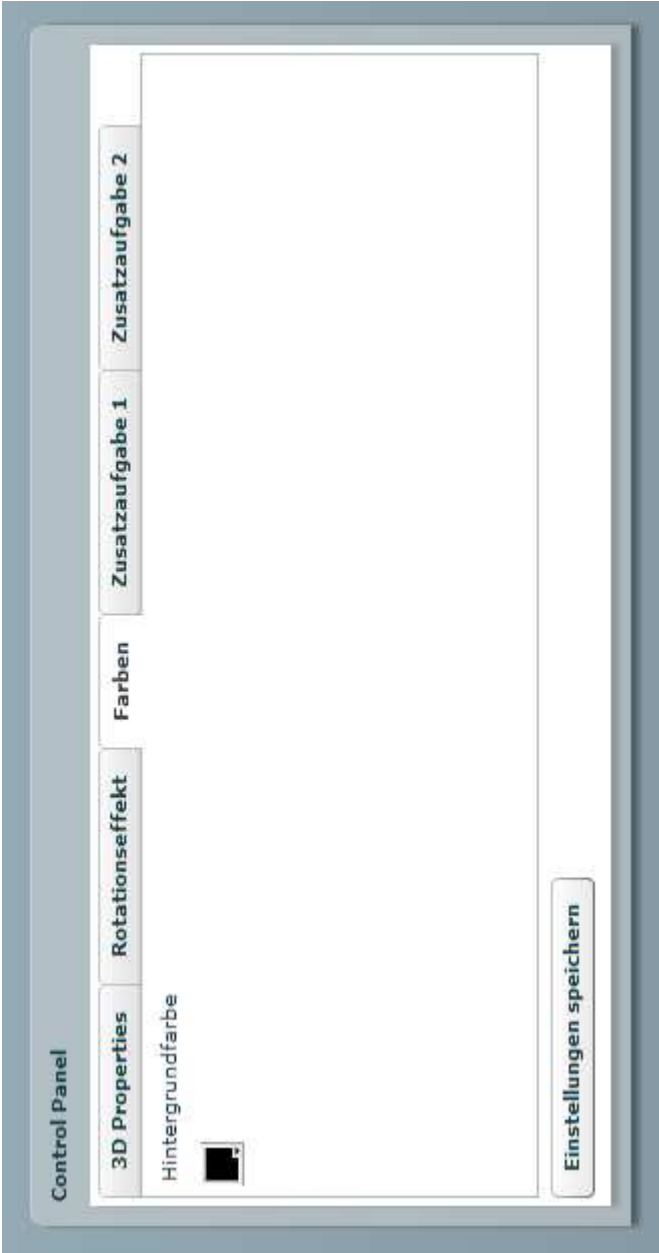
The screenshot shows a 'Control Panel' with several sections and sliders. The sections are: '3D Properties', 'Rotationseffekt', 'Farben', 'Zusatzaufgabe 1', and 'Zusatzaufgabe 2'. The sliders are: 'Field Of View', 'Projektionsmittelpunkt X', 'Projektionsmittelpunkt Y', 'X-Rotationswinkel', 'Y-Rotationswinkel', 'Z-Rotationswinkel', 'Z-Position', and 'Z-Position Playlist'. A 'Einstellungen speichern' button is at the bottom. Three callout boxes provide numerical values for specific sliders.

Slider Label	min	max	Wert
Field Of View	1	90	20
Projektionsmittelpunkt X	0	Applikationsbreite	$\max * 0.5$
Projektionsmittelpunkt Y	0	Applikationshöhe	$\max * 0.5$
X-Rotationswinkel	-90	90	0
Z-Position	-300	600	0

Screenshot 2:



Screenshot 3:



Daniel Bertram
Kerstin Götze

ASS Flex Workshop für die FH Köln, Campus
Gummersbach

16. Juni 2010

Tag 1 - Übung 5 – Bild & Font

Aufgabenstellung:

Fügt das Flex Logo (flexlogo.png) als eingebettetes Bild in die rechte obere Ecke der Anwendung ein.

Bettet außerdem die euch zur Verfügung gestellte Schriftart in die Anwendung ein. Setzt sie als Standardschriftart für die Anwendung.

Zusatzaufgabe(n):

Keine

Hinweise:

- Kopiert euch die Assets für Tag 1 Übung 5 vom Server und fügt sie in den assets-Ordner eures Flex Projektes ein.
- Für die jeden Schriftstil einer Schriftart (bold, italic, ...) muss jeweils die entsprechende Font-Datei eingebettet werden, die diesen Stil enthält.

Code Snippets:

```
// Eine Schriftart mit mx:Style einbinden. (Diese muß noch gesetzt werden.)
```

```
<mx:Style>
    @font-face {
        src: url("../assets/font/verdana.TTF");
        fontFamily: myFontFamily;
        advancedAntiAliasing: true;
        fontWeight: bold;
    }
</mx:Style>
```

```
// Bilder einbetten mit MXML
```

```
<mx:Image source="@Embed('Bildpfad')" />
```

```
// Bilder einbetten mit AS3
```

```
[Embed(source="icon.png")] private var icon:Class;

private function useImage():void{
    var imgObj:BitmapAsset = new icon() as BitmapAsset;
    var img:Image = new Image();
    img.source = imgObj;
}
```

Daniel Bertram
Kerstin Götze

ASS Flex Workshop für die FH Köln, Campus
Gummersbach

16. Juni 2010

Tag 2 - Übung 1 – Klassen, Getter und Setter

Aufgabenstellung:

Erzeugt eine MXML-Komponente „ControlPanel“. Diese soll von TitleWindow erben. Verschiebt alle GUI-Elemente des in den vorherigen Übungen erzeugten Fensters in diese Klasse.


Erzeugt eine weitere MXML-Komponente „MediaPlayer“, die von TitleWindow ableitet. Diese soll Getter und Setter für die Eigenschaft `_url` vom Typ String erhalten. Der Getter soll die aktuelle URL per `trace()` ausgeben. Fügt dem Fenster außerdem eine Instanz von VideoDisplay hinzu.

Erzeugt ein AS3-Klasse „Playlist“, die ebenfalls von TitleWindow ableitet. Setzt im Konstruktor Breite, Höhe und Titel des Fensters.

Zusatzaufgabe(n):

Keine.

Hinweise:

- Das `trace()` Statement wird für Debug-Ausgaben in die Console genutzt. Um diese Ausgaben zu sehen muss die Anwendung durch Klick des Buttons  kompiliert werden. Die Debug-Ausgaben tauchen dann in der Konsole des FlexBuilders auf
- Getter und Setter einer Eigenschaft, dürfen nicht genau den gleichen Namen haben, wie diese Eigenschaft, sonst entsteht bei Aufruf des Getters oder Setters eine Endlosschleife
- Getter und Setter werden in einem MXML Klasse in den Skript-Block geschrieben
- Auf Getter und Setter einer Klasse kann von außen wie auf eine `public var` zugegriffen werden. In der MXML-Syntax kann der Setter als Attribut genutzt werden (siehe Code Snippets)

Code Snippets:

```
// Getter und Setter
```

```
private var _volume:Number;
```

```
public function set volume(value:Number):void{  
    _volume = value;  
}
```

```
public function get volume():Number{  
    return _volume;  
}
```

```
// Eigene GUI Klassen in der Anwendung verwenden (MXML Syntax)
```

```
<mx:Application  
    xmlns:mx="http://www.adobe.com/2006/mxml"  
    xmlns:local="*"  
>  
    <local:MyClass />  
</Application>
```

```
// Eigene GUI Klassen in der Anwendung verwenden (AS3 Syntax)
```

```
private var createGui():void{  
    var myObject:MyClass = new MyClass();  
    this.addChild(myObject);  
}
```

```
// Auf Setter in MXML-Syntax zugreifen
```

```
<local:Player  
    volume="0.5"  
>
```

Daniel Bertram
Kerstin Götze

ASS Flex Workshop für die FH Köln, Campus
Gummersbach

16. Juni 2010

Tag 2 - Übung 2 – Events

Aufgabenstellung:

Fügt dem Control Panel folgendes Verhalten hinzu:

Button „Einstellungen speichern“:

- Klick: Alert erscheint mit Text „Einstellungen gespeichert“
- Mouse über Button: Schrift des Buttons wird kursiv
- Mouse verlässt Button: Schrift des Buttons wieder normal

ColorPicker:

- Bei Farbauswahl, Hintergrundfarbe der Anwendung auf die gewählte Farbe setzen


Slider „Field of View“, Projektionsmittelpunkt X/Y“:

- Bei Slider-Bewegung neuen Wert mit `trace()`-Statement ausgeben

Zusatzaufgabe(n):

- Über den „x“-Button in der Titelleiste des Control Panels, soll dieses geschlossen werden.

Hinweise:

- Schaut euch in der Online Doku an, welche Events die Komponenten werfen können.
- Schaut in der Online Doku nach, wie das TitleWindow mit einem Schließen-Button („x“-Button) versehen werden kann.
- Das trace() Statement wird für Debug-Ausgaben in die Console genutzt. Um diese Ausgaben zu sehen muss die Anwendung durch Klick des Buttons  kompiliert werden. Die Debug-Ausgaben tauchen dann in der Konsole des FlexBuilders auf
- An jeder beliebigen Stelle im Code kann über Application.application auf die Hauptapplikation zugegriffen werden.

Code Snippets:

```
// Event Listener werden in MXML so hinzugefügt:
```

```
<mx:Button label="click me" click="onButtonClick(event)"/>
```

```
// Event Listener werden in AS3 so hinzugefügt:
```

```
myButton.addEventListener(MouseEvent.CLICK, onButtonClick);
```

```
// So wird ein Style per AS3 gesetzt
```

```
guiObj.setStyle(„fontSize“, 12);
```

Daniel Bertram
Kerstin Götze

ASS Flex Workshop für die FH Köln, Campus
Gummersbach

16. Juni 2010

Tag 2 - Übung 3 – Audio & Video

Aufgabenstellung:

Erweitert die Klasse MediaPlayer so, dass der Setter für `_url` die Eigenschaft `source` für das `VideoDisplay` setzt.

Fügt auch folgende Buttons mit entsprechender Funktionalität ein:

- Einen 'Start' Button
- Einen 'Stop' Button
- Einen 'Pause' Button

Fügt einen Slider hinzu, der die Lautstärke des Videos regelt.

Zusatzaufgabe(n):

- Fügt einen weiteren Slider hinzu, mit dem ihr im Video navigieren könnt.

Hinweise:

- Kopiert euch die Assets für Tag 2 Übung 3 vom Server und fügt sie in den `assets`-Ordner eures Flex Projektes ein.
- Schaut euch die Klasse `VideoDisplay` in der Online Doku an. Achtet besonders auf die Events, die sie wirft.

Code Snippets:

Keine.

Daniel Bertram
Kerstin Götze

ASS Flex Workshop für die FH Köln, Campus
Gummersbach

16. Juni 2010

Tag 2 - Übung 4 – Styles & Skins

Aufgabenstellung:

Passt die Anwendung mittels externem CSS und Filtern folgendermaßen an:

- Hintergrundfarbe der Applikation: #ADCCA7
- Buttons allgemein: Farbe #7B9177, Alpha 1
 - Pause Button: Farbe #FFD91C
 - Stop Button: Farbe #FF361C
- Slider: slider.png als Skin für den Anfasser
- Mediaplayerbuttons: GlowFilter in ihrer jeweiligen Farbe

Zusatzaufgabe(n):

- Erstellt einen Programmatischen Skin für den Pause Button.
 - upSkin: Farbe #FFD91C
 - overSkin: Farbe #FFA14F
 - 2px breiter Rand in der Farbe #FF6100

Hinweise:

- Kopiert euch die Assets für Tag 2 Übung 4 vom Server und fügt sie in den assets-Ordner eures Flex Projektes ein.
- Filter fügt ihr mit <mx:filters> in MXML-Code ein
- Beachtet bitte, dass die Flex Syntax nicht zu 100% mit der CSS Syntax übereinstimmt !

Code Snippets:

```
// CSS Syntax
```

```
Button {  
    fontSize: 15;  
    fillColors: #FF0000, #FF0000;  
}  
.greenButtonStyle {  
    fontSize: 12;  
    fillColors: #00FF00, #00FF00;  
}
```

```
// Einbinden eines CSS mit
```

```
<mx:Style source="../../assets/Styles.css"/>
```

```
// Objekten eine CSS Style zuweisen:
```

```
guiObj.styleName="greenButtonStyle";
```

```
// Filter verwenden
```

```
<mx:Label id="label1" text="Filter Test" fontSize="20">  
    <mx:filters>  
        <mx:DropShadowFilter blurX="5" blurY="5" distance="20"/>  
    </mx:filters>  
</mx:Label>
```

Daniel Bertram
Kerstin Götze

ASS Flex Workshop für die FH Köln, Campus
Gummersbach

16. Juni 2010

Tag 2 - Übung 5 – Dateizugriff

Aufgabenstellung:

Fügt dem MediaPlayer einen Button „Video speichern“ hinzu.

Bei Klick auf diesen Button soll das Video von folgender URL heruntergeladen und gespeichert werden:

http://www.rockabyte.com/kgoetze/flex_workshop/demo.flv

Zusatzaufgabe(n):

- Fügt einen weiteren Button ein, der einen Dateidialog öffnet, der nur Dateien mit den Erweiterungen PNG, JPG und GIF anzeigt. Ladet ein beliebiges Bild als Hintergrundbild in eure Anwendung. Ihr könnt euch die Assets für Tag 2 Übung 5 herunterladen, darin findet ihr ein Bild zum Ausprobieren.

Hinweise:

- Schaut euch die Klasse FileReference in der Online Doku an.
- Zusatzaufgabe: Zunächst müssen die Bytes der gewählten Datei geladen werden. Diese müssen dann in ein Bitmap konvertiert werden.

Code Snippets:

Keine.

Daniel Bertram
Kerstin Götze

ASS Flex Workshop für die FH Köln, Campus
Gummersbach

16. Juni 2010

Tag 3 - Übung 1 – Dataprovider

Aufgabenstellung:

Erweitert die AS3 Klasse Playlist folgendermaßen:

- fügt ein List Objekt namens `_liste` ein
- fügt zwei globale Variablen `_listData` (ArrayCollection) und `_selectedURL` (String) hinzu und schreibt jeweils Getter und Setter für diese Variablen
- der Setter für `_listData` setzt den Data Provider von `_liste`

Beim Auftreten des `creationComplete` Events der Anwendung, soll die Eigenschaft `_listData` der Playlist mit Demodaten gefüllt werden. Kopiert euch dazu die Assets für Tag 3 Übung 1 vom Server und fügt sie in den `assets`-Ordner eures Flex Projektes ein. Füllt den `_listData` mit den Pfaden der Videos.

Bei Klick auf einen Playlistertrag soll die Eigenschaft `_selectedURL` der Playlist auf den angeklickten Eintrag gesetzt werden. Bindet die `_url` des `MediaPlayers` an `_selectedURL` der Playlist.

Zusatzaufgabe(n):

- Fügt mit Hilfe des Data Providers den ersten zwei Playlist-Einträgen ein Icon hinzu. Die Icons findet ihr in `assets/img`

Hinweise:

- Um ein DataBinding durchzuführen müssen für die zu bindende Variable ein Getter und Setter existieren, die mit dem [Bindable]-Tag gekennzeichnet sind.
- Nur eingebettete Bilder können als Listenicons verwendet werden

Code Snippets:

```
// Binden der Daten eines TextInputs an ein Text Objekt:
```

```
<mx:TextInput id="myTI"/>           <!--source-->  
<mx:Text id="myText" text="{myTI.text}"/>   <!--target-->
```

```
// Füllen einer Liste mit Daten und Auslesen von Daten aus dem  
selektierten Listeneintrag
```

```
<mx>List id="list" width="400" itemClick="onItemClick()"/>  
<mx:TextArea id="output"/>
```

```
<mx:Script> <![CDATA[  
    import mx.events.ListEvent;  
    import mx.collections.ArrayCollection;  
  
    private var dp:ArrayCollection;  
  
    private function initDataProvider():void{  
        dp = new ArrayCollection();  
        dp.addItem({label:"Google", url:"www.google.de"});  
        dp.addItem({label:"Yahoo", url:"www.yahoo.de"});  
        list.dataProvider = dp;  
    }  
  
    private function onItemClick():void{  
        output.text = list.selectedItem.url as String;  
    }  
    ]]></mx:Script>
```

Daniel Bertram
Kerstin Götze

ASS Flex Workshop für die FH Köln, Campus
Gummersbach

16. Juni 2010

Tag 3 - Übung 2 – Effekte

Aufgabenstellung:

Verseht die Komponenten der Anwendung mit folgenden Effekten:

- Klick auf Pause Button: ein Fade Effekt lässt VideoDisplay innerhalb von 0,5 Sekunden halb transparent werden.
- Klick auf Start Button: Fade Effekt wieder rückgängig

Zusatzaufgabe(n):

MediaPlayer-Komponente und Playlist-Komponente:

- Cursor über TitleWindow: Fenster erhält Glow Effekt und der Rand des Fensters wird komplett undurchsichtig
- Cursor verlässt TitleWindow: vorherige Effekte rückgängig

Hinweise:

- Auch Effekte lösen Events aus.
- Effekte sind in Flex als eigene Klasse geschrieben, die ihr unter *mx.effects* finden könnt.
- Schaut euch die Klasse *AnimateProperty* einmal genauer an
- Es gibt spezielle Effekte, die dafür sorgen, dass mehrere Effekte gleichzeitig oder aufeinanderfolgend abgespielt werden

Code Snippets:

```
// Erstellen eines Effektes und auslösen durch einen Button:
```

```
<mx:WipeLeft id="myWL" duration="1000" target="{myPanel}"/>
```

```
<mx:Button label="Play effect" click="{myWL.play()}" />
```

```
// Ein Objekt per AS3 automatisch einen Effekt auslösen lassen:
```

```
object.setStyle(„effectName“, effectInstance);
```

Daniel Bertram Kerstin Götze	ASS Flex Workshop für die FH Köln, Campus Gummersbach 16. Juni 2010
---------------------------------	---

Tag 3 - Übung 3 – 3D Support

Aufgabenstellung:

Fügt der Applikation in einer `init()`-Methode des `ControlPanel`s eine neue `PerspectiveProjection` hinzu. Setzt die Initialwerte der Projektion auf die Initialwerte der Slider „Field of View“, „Projektionsmittelpunkt X“ und „Projektionsmittelpunkt Y“.

Fügt dem `ControlPanel` folgende Funktionalität hinzu:

- Field of View Slider: verändert `fieldOfView` der Projektion
- Projektionsmittelpunkt X und Y Slider: verändern den Mittelpunkt der Projektion
- X-, Y- und Z-Rotationswinkel Slider: Verändern X-, Y- und Z-Rotation von `MediaPlayer` und `Playlist`
- Z-Position Slider: verändern Z-Position von `MediaPlayer` und `Playlist`

Wenn sich die URL des abzuspielenden Videos im `MediaPlayer` ändert, soll dieser sich um 360 Grad um die X-Achse drehen.

Zusatzaufgabe(n):

Fügt dem Tab „Rotationseffekt“ im ControlPanel folgende Funktionalität hinzu:

- NumericStepper „Dauer“: stellt die Dauer des Effekts ein
- Radiobuttons „Achse“: setzen die Drehachse des Effekts
- Checkbox: aktiviert/deaktiviert den Effekt

Hinweise:

- arbeitet mit DataBinding
- GUI-Komponenten, die mit MXML-Coder erstellt wurden, sind nach außen als public Objekte sichtbar
- Auf die Hauptapplikation kann jederzeit mit Application.application zugegriffen werden
- Änderungen des Field of View haben nur dann einen sichtbaren Effekt, wenn die Z-Position der GUI Objekte ungleich 0 ist

Code Snippets:

```
// PerspectiveProjection erstellen  
var pp:PerspectiveProjection = new PerspectiveProjection();  
pp.fieldOfView = 20;  
pp.projectionCenter = new Point(0, 0);  
this.transform.perspectiveProjection = pp;
```