MovieBench – Anleitung für Autoren

MovieBench ermöglicht die Entwicklung und Distribution interaktiver Videoclip-basierter Lernmodule.
Das Design von MovieBench kann weitgehend den eigenen Bedürfnissen angepasst werden. Dazu sind keinerlei Programmierkenntnisse notwendig. Vorausgesetzt ist lediglich der geübte Umgang mit einem Grafikprogramm (Photoshop, Illustrator, PaintShop, CorelDraw) – und etwas gestalterisches Flair. Bild 1 & 2 zeigen zwei unterschiedliche Designs. Position, Grösse und Aussehen der unterschiedlichen Komponenten sind frei wählbar!

Bild1

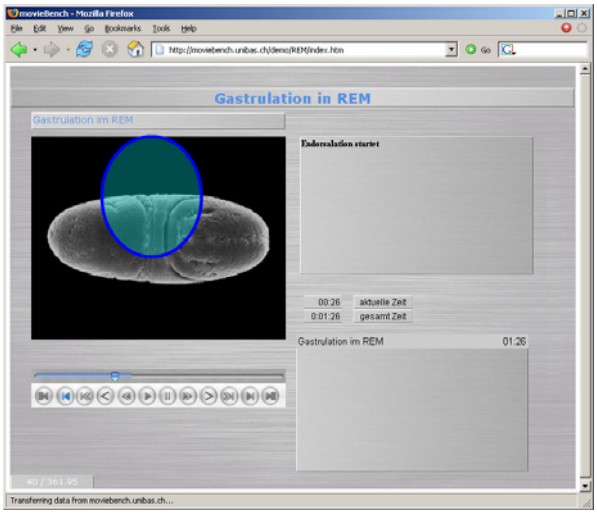


Bild 2

Die Interaktivität beschränkt sich nicht auf die Standard Videocontrols, sondern kann mit sogenannten Hotspots erweitert werden. Hotspots sind zeitlich und örtlich begrenzte Bereiche innerhalb eines Videoclips. Hotspots ermöglichen Interaktion. Beispiele dafür sind das Anzeigen von Erläuterungen in einer Textbox, das Öffnen eines Browserfensters mit einer bestimmten Webseite, oder der Sprung zu einem anderen Videoclip. Solche Aktionen können automatisch oder auf Reaktion der Benutzerin erfolgen. Auch für die Erstellung der Interaktionen werden keinerlei Programmierkenntnisse benötigt. Bild 2 zeigt einen ovalen, semitransparenten Hotspot.

Diese Anleitung beschreibt Schritt für Schritt, wie Sie mit MovieBench Ihr eigenes Lernmodul entwickeln. Kapitel 1 zeigt einer Übersicht der beteiligten Dateien und der Verzeichnisstruktur. In Kapitel 2 werden die das Layout bestimmenden Komponenten vorgestellt. Wie Sie in einer Playlist die Abspielfolge der Videoclips und die Interaktionen definieren, erfahren Sie in Kapitel 3. Einige Tipps finden Sie in Kapitel 4 und Kapitel 5 verweist auf weiterführende Literatur.

Viel Spass bei der Erstellung Ihres ersten Lernmoduls wünscht Ihnen das URZ-LTN Team!

Inhaltsverzeichnis

1. Dateien und Verzeichnisstruktur

Ein MovieBench Lernmodul besteht immer aus einer Sammlung von Dateien, die der Übersicht halber in einer anschaulichen Verzeichnisstruktur abgelegt sind.

1.1. Die Dateien

1.1.1. HTML-Datei

Dies ist die Datei, die vom Benutzer im Webbrowser aufgerufen wird. In der Regel heisst diese Datei index.html.

1.1.2. moviebench.swf

Der eigentliche MovieBench Player. Dieser ist eine in Macromedia Flash MX 2004 mit Actionscript 2.0 geschriebene Applikation, welche in der aufgerufenen HTML-Datei eingebettet ist.

1.1.3. XML-Dateien

Diese beschreiben die Konfiguration des MovieBench Lernmoduls.

1.1.3.1. configuration.xml

configuration.xml wird von moviebench.swf aufgerufen und verweist auf die zwei Dateien, in denen das Layout und die Playlist beschrieben werden.

```
<?xml version="1.0" encoding="utf-8"?>
<Configuration>
    <Layout source="xml/layout.xml" />
    <Playlist source="xml/playlist.xml" />
</Configuration>
```

Der Pfad zu den XML-Dateien liegt relativ zur HTML-Datei index.html.

1.1.3.2. layout.xml

In dieser XML-Datei wird das Aussehen des MovieBench Lernmoduls beschrieben. Also Grösse, Position und Aussehen der einzelnen Komponenten.

1.1.3.3. playlist.xml

Enthält eine Liste der zum Lernmodul gehörenden Videoclips, sowie zu jedem Videoclip eine Liste dazugehöriger Hotspots.

1.1.4. Layout-Dateien

Das Aussehen der einzelnen Komponenten von MovieBench wird durch Bilder im Format JPEG (z.Bsp. hintergrund.jpg, play-n.jpg) bestimmt.

1.1.5. Videoclips

Die eigentlichen Videoclips müssen alle in das Flash Videoformat (meinclip.flv) konvertiert werden. Eine einfache und meist ausreichende Konvertierungsmöglichkeit bietet der kostenlose Riva Encoder. Sorenson Squeeze ist eine professionelle, aber kostenpflichtige Alternative.

1.2. Die Verzeichnisstruktur

Die nachfolgend beschriebene Dateistruktur ist lediglich eine Empfehlung und nicht zwingend. Sie geht davon aus, dass Sie mehrere Lernmodule erstellen.

MeineLernmodule/
- index.html *(enthält die Liste aller meiner Lernmodule)*
- moviebench.swf

MeinErstesLernmodul/
- index.html *(HTML-Datei für «MeinErstesLernmodul»)*
- **layout/** *(alle, das Aussehen von MovieBench bestimmenden jpg-Dateien)*
  - hintergrund.jpg
  - playhead.jpg
  - playheadbox.jpg
  - playheadprogress.jpg
  - … etc.
  - **scrollbar/** *(jpg für den Scrollbar liegen in Unterverzeichnis)*
    - scrollbar.jpg
    - scrollthumb.jpg
    - … etc.
  - **vcbtn/** *(jpg für die Videocontrol liegen in Unterverzeichnis)*
    - play-n.jpg *(jpg für Play-Taste im Status Normal)*
    - play-o.jpg *(jpg für Play-Taste im Status MouseOver)*
    - play-d.jpg *(jpg für Play-Taste im Status MouseDown)*
    - next-n.jpg
    - … etc.
- **videos/**
  - ErsterClip.flv
  - ZweiterClip.flv
  - DritterClip.flv
  - … etc.
- **xml/**
  - configuration.xml *(enthält lediglich den Verweis auf die anderen XML-Dateien)*
  - layout.xml *(beschreibt Position, Grösse und Aussehen der Komponenten)*
  - playlist.xml *(Liste der Videoclips und ihrer Hotspots)*

MeinZweitesLernmodul/
- …

1.3. HTML-Datei

Beispiel einer HTML-Datei (z.Bsp. MeinErstesLernmodul/index.html), welche moviebench.swf und die zugehörige Konfigurationsdatei configuration.xml aufruft.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
          "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">

<head>
  <style type= "text/css">
    html, body {height: 100%; margin: 0; padding: 0; vertical-align:middle;}
  </style>

  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
  <title>movieBench</title>
</head>

<body bgcolor="#ffffff"
      leftmargin="0" rightmargin="0" topmargin="0" bottommargin="0">

  <div align="center" style="width:100%; height:100%;  ">

    <object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"
            codebase="http://fpdownload.macromedia.com/pub/shockwave/cabs/~~
              ~~flash/swflash.cab#version=7,0,0,0" width="100%" height="100%"
            id="movieBench">
      <param name="allowScriptAccess" value="sameDomain" />
      <param name="movie" value="../moviebench.swf?config=xml/configuration.xml" />
      <param name="bgcolor" value="#333333" />
      <PARAM NAME="scale" VALUE="exactfit"/>
      <param name="menu" value="false" />

      <embed src="../moviebench.swf?config=xml/configuration.xml" menu="false"
             scale="exactfit" bgcolor="#333333" width="100%" height="100%"
             name="movieBench" allowScriptAccess="sameDomain"
             type="application/x-shockwave-flash"
             pluginspage="http://www.macromedia.com/go/getflashplayer" />
    </object>

  </div>
</body>

</html>
```

Der mit «~~ ~~» getrennte URL ist auf einer Zeile und ohne Leerzeichen dazwischen zu schreiben.
Der Pfad zu moviebench.swf und configuration.xml liegt relativ zum Standort der HTML-Datei index.html.

2. Layout

2.1. Das Prinzip der Komponenten

Das Layout eines MovieBench Lernmoduls wird in der XML-Datei **layout.xml** beschrieben. Die Komponenten sind hierarchisch angeordnet. Die alles beinhaltende Komponente ist **UIDesktop**. Diese beinhaltet z.B. den Preloader (eine Komponente vom Typ **UIImage** mit der id **preloader**) und das Hintergrundbild (ebenfalls eine Komponente vom Typ **UIImage**, diesmal mit der id **background**). Komponenten innerhalb von **UIDesktop** können weitere Komponenten beinhalten. Die Liste der Videoclips **UIList** beinhaltet wiederum ein Hintergrundbild **UIImage** sowie den Scrollbar **UIScrollbar**, der seinerseits seine Teilkomponenten, wie beispielsweise die Auf- und Ab-Buttons enthält.

Im Folgenden sind alle Komponenten aufgelistet, die innerhalb UIDesktop möglich sind:

- UIButton
- UICurrentTime
- UIImage
- UILabel
- UIList
- UIMediaController
- UIMediaDisplay
- UIMuteButton
- UIPlayhead
- UIPosition
- UIProgressBar
- UIScrollBar
- UITaskBar
- UITextArea
- UITimeLine
- UITitle
- UITotalTime

Beachte:Die Schreibweise innerhalb der XML-Dateien ist casesensitive. D.h. wenn Sie UITextarea statt UITextArea schreiben, wird MovieBench nicht funktionieren. Da keine Fehlermeldungen angezeigt werden, empfiehlt sich sorgfältiges Arbeiten.

2.2. Die Komponenten-Hierarchie

Die folgende Liste zeigt eine typische Komponenten-Hierarchie. Jede Komponente benötigt eine auf ihrer Hierarchiestufe eindeutige **id**. Diese ist in Klammer dargestellt.

- UIDesktop (desktop)
  - UIImage (preloader)
  - UIImage (background)
  - UITaskBar (taskBar)
  - UITitle (title)
    - UIImage (background)
  - UIMediaDisplay (master)
    - UIImage (preloader)
  - UIMediaDisplay (slave)
    - UIImage (preloader)
  - UIMediaController (mediaController)
    - UIImage (background)
    - UITimeLine (timeLine)
      - UIImage (background)
      - UIPlayhead (playhead)
      - UIProgressBar (progressBar)
    - UIButton (play)
    - UIButton (pause)
    - *weitere UIButton-Komponenten (die restlichen Bezeichner für MediaControl-Schaltflächen)*
  - UIList (list)
    - UIImage (background)
    - UIScrollBar (scrollBar)
      - UIImage (scrollTrack)
      - UIButton (scrollThumb)
      - UIButton (scrollUpArrow)
      - UIButton (scrollDownArrow)
  - UITextArea (textArea)
    - UIImage (background)
    - UIScrollBar (scrollBar)
      - UIImage (scrollTrack)
      - UIButton (scrollThumb)
      - UIButton (scrollUpArrow)
      - UIButton (scrollDownArrow)
  - UITotalTime (totalTime)
    - UIImage (background)
  - UICurrentTime (currentTime)
    - UIImage (background)
  - UIPosition (position)
    - UIImage (background)
  - UIMuteButton (masterAndSlaveBtn | masterMuteBtn | slaveMuteBtn)
  - UILabel (irgendwas)

2.3. Komponenten und Attribute

Die grafischen Oberflächenelemente basieren ebenfalls auf ActionScript 2.0 und wurden als (Flash MX-) Komponenten angelegt. Das Layout der Komponenten bestimmt eine XML basierte Layout-Datei. Das Aussehen der Komponenten wird durch jpeg Bilder bestimmt. Das Attribut `depth:` bestimmt die Darstellungstiefe (Ebene/Layer), in der die Komponente liegt. Eine Komponente mit der Tiefe 8 liegt über einer anderen Komponente mit der Tiefe 7. Komponenten der gleichen Hierarchiestufe müssen zwingend unterschiedliche Tiefen (`depth:`) besitzen.
Attribute werden immer in der Form key="value" angegeben, wobei value immer in doublequotes geschrieben wird und entweder vom Typ String, Number oder Boolean ist.

2.3.1. UIButton

Die Komponente UIButton dient als Schaltfläche und wird innerhalb anderer Komponenten (z.B. UIMediaController, UIScrollBar) als interaktives Element eingesetzt.

2.3.1.1. Parameter

`id:String`: Instanzname in Flash (muss angegeben werden);

`depth:Number`: Darstellungstiefe in der Stapelreihenfolge (muss angegeben werden);

`x:Number:` Horizontale Position, die immer relativ zum übergeordneten Element angegeben wird (optional, Standardwert ist 0);

`y:Number:` Vertikale Position, die immer relativ zum übergeordneten Element angegeben wird (optional, Standardwert ist 0);

`width:Number:` Breite (optional, andernfalls wird die Breite automatisch berechnet);

`height:Number`: Höhe (optional, andernfalls wird die Breite automatisch berechnet);

`upSource:`String: Pfadangabe für das Jpeg-Bild des Normalzustandes (muss angegeben werden);

`overSource:String`: Pfadangabe für das Jpeg-Bild des Darüberzustandes (optional, andernfalls wird der Normalzustand angezeigt);

`downSource:String`: Pfadangabe für das Jpeg-Bild des Gedrücktzustandes (optional, andernfalls wird der Normalzustand angezeigt);

2.3.1.2. Beispiel

```
<UIButton depth="2" id="scrollUpArrow" x="0" y="174" width="20" height="20"
upSource="layout/up.jpg" overSource="layout/over.jpg"/>
```

2.3.2. UICurrentTime

Die Komponente UICurrentTime zeigt den nummerischen Wert des Abspielkopfes innerhalb des aktuellen Videoclips in der Form mm:ss.ttt (Minuten, Sekunden, Ticks für Millisekunden) an.

2.3.2.1. Parameter

`id:String`: Instanzname in Flash (muss angegeben werden);

`depth:Number`: Darstellungstiefe in der Stapelreihenfolge (muss angegeben werden);

`x:Number:` Horizontale Position, die immer relativ zum übergeordneten Element angegeben wird (optional, Standardwert ist 0);

`y:Number:` Vertikale Position, die immer relativ zum übergeordneten Element angegeben wird (optional, Standardwert ist 0);

`width:Number:` Breite (optional, andernfalls wird die Breite automatisch berechnet);

`height:Number:` Höhe (optional, andernfalls wird die Breite automatisch berechnet);

`font:String:` Schriftart (optional, Standard ist Verdana);

`fontSize:Number:` Schriftgröße (optional, Standard ist 12);

`fontColor:Number:` Schriftfarbe (optional, Standard ist 0x000000 für Schwarz);

`backgroundColor:Number:` Hintergrundfarbe (optional, falls eine Hintergrundgrafik angegeben ist, wird dieser Parameter übergangen);

`bold:Boolean:` Schriftstil-Fett (optional);

`italic: Boolean:` Schriftstil-Kursiv (optional);

`underline: Boolean:` Schriftstil-Unterstrichen (optional);

Eine Hintergrundgrafik wird als untergeordnete UIImage-Komponente mit der id „background“ angegeben – die Positionierung und Größe wird automatisch der Texfeldgröße angepasst. Eine Hintergrundfarbe wird bei gleichzeitiger Angabe einer Hintergrundgrafik ignoriert.

2.3.2.2. Beispiel

```
<UICurrentTime id="currentTime" depth="10" x="415" y="325" width="60" height="20"
font="Arial, Helvetica" fontSize="12" fontColor="0xFF0000"
backgroundColor="0xFFFF00" bold="false" italic="true" underline="false" />
```

2.3.3. UIDesktop

Diese Komponente ist der Container, in dem die anderen Komponenten platziert werden. Aus diesem Grund muss diese Komponente zwingend als oberstes, resp. äusserstes Element in der Layout-Beschreibung angegeben sein.

2.3.3.1. Parameter

`id:String`: Instanzname in Flash (muss angegeben werden);

`depth:Number`: Darstellungstiefe in der Stapelreihenfolge (muss angegeben werden);

`width:Number:` Breite (optional, andernfalls wird die Breite automatisch berechnet);

`height:Number:` Höhe (optional, andernfalls wird die Breite automatisch berechnet);

`quality:String`: Optionale Angabe der Darstellungsqualität in der Form „`low`“, „`middle`“ oder „`high`“ (Standard ist „`low`“, welches sich gerade bei der Verwendung von Bitmaps empfiehlt, um Weichzeichnungsartefakte zu verhindern);

2.3.3.2. Beispiel

```
<UIDesktop id ="desktop" depth="0" width="800" height="600" quality="high">
      <!-- Hier platzieren Sie die gewünschten Komponenten -->
</UIDesktop>
```

2.3.4. UIImage

Diese Komponente wird zur Darstellung von Bildelementen eingesetzt. Zum einen dient sie als Hintergrund und zum anderen auch zur Anzeige von Bildelementen in anderen Komponenten.

2.3.4.1. Parameter

`id:String`: Instanzname in Flash (muss angegeben werden);

`depth:Number`: Darstellungstiefe in der Stapelreihenfolge (muss angegeben werden);

`x:Number:` Horizontale Position, die immer relativ zum übergeordneten Element angegeben wird (optional, Standardwert ist 0);

`y:Number:` Vertikale Position, die immer relativ zum übergeordneten Element angegeben wird (optional, Standardwert ist 0);

`width:Number:` Breite (optional, andernfalls wird die Breite automatisch berechnet);

`height:Number:` Höhe (optional, andernfalls wird die Breite automatisch berechnet);

`source:String`: Pfadangabe für das Jpeg-Bild (muss angegeben werden);

2.3.4.2. Beispiel

```
<UIImage id="scrollTrack" depth="0" source="layout/menusliderbox.jpg" />
```

Spezielle UIImage-Komponenten ***Hinweis***

Generell wird diese Komponente mit der Bezeichnung „background“ verwendet, um Hintergrundgrafiken zu initialisieren (`id="background"`). Es können aber auch beliebige Bilder im Desktop platziert werden. In wenigen Fällen wird diese Komponente jedoch auch zur Initialisierung spezieller Elemente benötigt:

- `id="preloader"` zur Bestimmung eines Preloaders im Gesamtlayout;
- `id="preloader"` zur Angabe eines Preloaders für das Vorausladen eines Videos in der UIMediaDisplay-Komponente;
- `id="scrollTrack"` definiert innerhalb der UIScrollBar den Rollbalken;

2.3.5. UILabel

Die Komponente UILabel stellt einzeilige statische Texte dar.

2.3.5.1. Parameter

`id:String`: Instanzname in Flash (muss angegeben werden);

`depth:Number`: Darstellungstiefe in der Stapelreihenfolge (muss angegeben werden);

`x:Number:` Horizontale Position, die immer relativ zum übergeordneten Element angegeben wird (optional, Standardwert ist 0);

`y:Number:` Vertikale Position, die immer relativ zum übergeordneten Element angegeben wird (optional, Standardwert ist 0);

`width:Number:` Breite (optional, andernfalls wird die Breite automatisch berechnet);

`height:Number:` Höhe (optional, andernfalls wird die Breite automatisch berechnet);

Der eigentliche Wert dieser Komponente wird als Knotenwert im XML angegeben. Einfaches HTML 1.0 ist zur Formatierung erlaubt (siehe Tipps). Die Angabe in einem CDATA-Block (`<![CDATA[ … ]]>`) ist dabei zu empfehlen!

Eine optionale Hintergrundgrafik wird als untergeordnete UIImage-Komponente mit der id „background“ angegeben;

2.3.5.2. Beispiel

```
<UILabel depth="1" id="myTitle" x="0" y="20" width="800" height="120"
border="false" selectable="false" background="true" backgroundColor="0xF0F0F0"
autoSize="left"><![CDATA[<html><P ALIGN="CENTER"><FONT FACE="Arial" SIZE="21"
COLOR="#FF0000"><B>TEST-Titel</B></FONT></P></html>]]>
  <UIImage id="background" depth="1" source="layout/box.jpg"/>
</UILabel>
```

Hinweis

Besonderheiten

Diese Komponente erlaubt zusätzlich auch alle Eigenschaften der Flash Klasse TextField. Um die Hintergrundfarbe definieren zu können, muss wie in Flash zuvor die Eigenschaft background auf true gesetzt werden und anschließend die Hintergrundfarbe über backgroundColor eingestellt werden. Falls die Breite und Höhe nicht definiert sind, wird der Text nach links laufen (`autosize = "left"`) und die Texfeldgröße passt sich automatisch an – außer diese Eigenschaft ist explizit deaktiviert bzw. ist nicht angegeben worden (siehe „Tipps“)!

2.3.6. UIList

Die UIList stellt einzelne Abschnitte der Playlist inkl. der jeweiligen Videoclip-Zeit dar. Links ist der Titel und rechts die Zeit im Format mm:ss sichtbar.

2.3.6.1. Parameter

`font:String`: Schriftart (optional, Standard ist "Verdana");

`fontSize:Number`: Schriftgröße unselektierter Listeneinträge (optional, Standard ist 12);

`fontColor:Number`: Schriftfarbe unselektierter Listeneinträge (optional, Standard ist 0x000000 für Schwarz);

`backgroundColor:Number`: Hintergrundfarbe unselektierter Listeneinträge (optional);

`selectedFontSize:Number`: Schriftgröße selektierter Listeneinträge (optional, Standard ist 13);

`selectedFontColor:Number`: Schriftfarbe selektierter Listeneinträge (optional, Standard ist 0x000000 für Schwarz);

`selectedBackgroundColor:Number`: Hintergrundfarbe selektierter Listeneinträge (optional);

`id:String`: Instanzname in Flash (muss angegeben werden);

`depth:Number`: Darstellungstiefe in der Stapelreihenfolge (muss angegeben werden);

`x:Number:` Horizontale Position, die immer relativ zum übergeordneten Element angegeben wird (optional, Standardwert ist 0);

`y:Number:` Vertikale Position, die immer relativ zum übergeordneten Element angegeben wird (optional, Standardwert ist 0);

`width:Number:` Breite (optional, andernfalls wird die Breite automatisch berechnet);

`height:Number:` Höhe (optional, andernfalls wird die Breite automatisch berechnet);

`bold:Boolean:` Schriftstil-Fett (optional);

`italic: Boolean:` Schriftstil-Kursiv (optional);

`underline: Boolean:` Schriftstil-Unterstrichen (optional);

Diese Komponente besitzt optional als untergeordnetes Element ein UIImage mir der id „background“ für den Hintergrund;

Diese Komponente besitzt optional als untergeordnetes Element ein UIScrollBar mit der id „scrollBar“ für den Rollbalken;

2.3.6.2. Beispiel

```
<UIList depth="8" id="menu" x="405" y="380" width="353" height="195"
selectedBackgroundColor="0xCCCCCC">
   <UIImage id="background" depth="0" width="328" source="layout/box.jpg"/>
</UIList>
```

2.3.7. UIMediaController

Die Komponente UIMediaController enthält die Steuerelemente für die Komponente UIMediaDisplay.

2.3.7.1. Parameter

`id:String`: Instanzname in Flash (muss angegeben werden);

`depth:Number`: Darstellungstiefe in der Stapelreihenfolge (muss angegeben werden);

`x:Number:` Horizontale Position, die immer relativ zum übergeordneten Element angegeben wird (optional, Standardwert ist 0);

`y:Number:` Vertikale Position, die immer relativ zum übergeordneten Element angegeben wird (optional, Standardwert ist 0);

`width:Number:` Breite (optional, andernfalls wird die Breite automatisch berechnet);

`height:Number:` Höhe (optional, andernfalls wird die Breite automatisch berechnet);

Als untergeordnete Schaltflächen sind UIButton-Komponenten mit den folgenden ids erlaubt: play; pause; fastForward; rewind; singleStepForward; singleStepBack; jumpToStartCurrentClip; jumpToEndCurrentClip; jumpToStartPreviousClip; jumpToStartNextClip; jumpToStartFirstClip und jumpToStartLastClip.

2.3.7.2. Beispiel

```
<UIMediaController depth="3" id="mediaController" x="30" y="325">
   <UIImage id="background" depth="0" x="0" y="0" source="layout/vcbox.jpg"/>
   <UIButton depth="1" id="play" x="149" y="2" upSource="layout/vcbtn/play_n.jpg"
overSource="layout/vcbtn/play_o.jpg" downSource="layout/vcbtn/play_d.jpg" />
   <UIButton depth="2" id="pause" x="178" y="2" upSource="layout/vcbtn/pause_n.jpg"
overSource="layout/vcbtn/pause_o.jpg" downSource="layout/vcbtn/pause_d.jpg" />
   <UIButton depth="3" id="fastForward" x="207" y="2"
upSource="layout/vcbtn/fwd_n.jpg" overSource="layout/vcbtn/fwd_o.jpg"
downSource="layout/vcbtn/fwd_d.jpg" />
   <UIButton depth="4" id="rewind" x="120" y="2" upSource="layout/vcbtn/rwd_n.jpg"
overSource="layout/vcbtn/rwd_o.jpg" downSource="layout/vcbtn/rwd_d.jpg" />
   <UIButton depth="5" id="singleStepForward" x="236" y="2"
upSource="layout/vcbtn/sff_n.jpg" overSource="layout/vcbtn/sff_o.jpg"
downSource="layout/vcbtn/sff_d.jpg" />
   <UIButton depth="6" id="singleStepBack" x="91" y="2"
upSource="layout/vcbtn/sfb_n.jpg" overSource="layout/vcbtn/sfb_o.jpg"
downSource="layout/vcbtn/sfb_d.jpg" />
   <UIButton depth="7" id="jumpToStartCurrentClip" x="62" y="2"
upSource="layout/vcbtn/prev_n.jpg" overSource="layout/vcbtn/prev_o.jpg"
downSource="layout/vcbtn/prev_d.jpg" />
   <UIButton depth="8" id="jumpToEndCurrentClip" x="265" y="2"
upSource="layout/vcbtn/next_n.jpg" overSource="layout/vcbtn/next_o.jpg"
downSource="layout/vcbtn/next_d.jpg" />
   <UIButton depth="9" id="jumpToStartPreviousClip"      x="33" y="2"
upSource="layout/vcbtn/start_n.jpg" overSource="layout/vcbtn/start_o.jpg"
downSource="layout/vcbtn/start_d.jpg" />
```

```
      <UIButton depth="10" id="jumpToStartNextClip" x="294" y="2"
   upSource="layout/vcbtn/end_n.jpg" overSource="layout/vcbtn/end_o.jpg"
   downSource="layout/vcbtn/end_d.jpg" />
      <UIButton depth="11" id="jumpToStartFirstClip" x="4" y="2"
   upSource="layout/vcbtn/first_n.jpg" overSource="layout/vcbtn/first_o.jpg"
   downSource="layout/vcbtn/first_d.jpg" />
      <UIButton depth="12" id="jumpToStartLastClip" x="323" y="2"
   upSource="layout/vcbtn/last_n.jpg" overSource="layout/vcbtn/last_o.jpg"
   downSource="layout/vcbtn/last_d.jpg" />
</UIMediaController>
```

2.3.8. UIMediaDisplay

Die Komponente UIMediaDisplay zeigt die Videoclips. Es existieren Anwendungen, in denen zwei Videos synchron abgespielt werden. Deshalb existiert optional eine zweite Videobox („slave“). Bitte beachten Sie, dass aber immer nur eine UIMediaControl-Komponente erlaubt ist, die beide UIMediaDisplays synchron steuert. Hotspots werden dabei immer nur im „master“ UIMediaDisplay gezeigt.

2.3.8.1. Parameter

`id:String`: Instanzname in Flash. Zur Wahl stehen „master“ für die primäre Videoanzeige und „slave“ für die sekundäre Videoanzeige (muss angegeben werden);

`depth:Number`: Darstellungstiefe in der Stapelreihenfolge (muss angegeben werden);

`x:Number:` Horizontale Position, die immer relativ zum übergeordneten Element angegeben wird (optional, Standardwert ist 0);

`y:Number:` Vertikale Position, die immer relativ zum übergeordneten Element angegeben wird (optional, Standardwert ist 0);

`width:Number:` Breite (optional, andernfalls wird die Breite automatisch berechnet, was jedoch aufgrund häufig fehlender Meta-Daten in Videoclips zu einer fehlerhaften Darstellung führen kann);

`height:Number:` Höhe (optional, andernfalls wird die Breite automatisch berechnet, was jedoch aufgrund häufig fehlender Meta-Daten in Videoclips zu einer fehlerhaften Darstellung führen kann);

Eine optionale Hintergrundgrafik wird als untergeordnete UIImage-Komponente mit der id „background“ angegeben;

Eine optionale Preloadgrafik wird als untergeordnete UIImage-Komponente mit der id „preloader“ angegeben;

2.3.8.2. Beispiel

```
<UIMediaDisplay depth="16" id="master" x="405" y="100" width="353" height="200">
   <UIImage id="background" depth="1" source="layout/box.jpg"/>
   <UIImage id="preloader" depth="2" x="80" y="30" source="layout/preloader.swf"/>
</UIMediaDisplay>
```

2.3.9. UIMuteButton

Die Komponente UIMuteButton dient als Schaltfläche zum Ein- und Ausschalten des gesamten Tons. Werden zwei UIMediaDisplays (master und slave) verwendet, dann kann der Ton auch einzeln geschaltet werden.

2.3.9.1. Parameter

`id:String`: Instanzname in Flash (muss angegeben werden);

`muteMode`:String: Optionales Ziel für die Tonsteuerung (erlaubte werte sind „masterMute“, „slaveMute“ und „masterAndSlaveMute“, Standard ist masterAndSlaveMute“);

`depth:Number`: Darstellungstiefe in der Stapelreihenfolge (muss angegeben werden);

`x:Number:` Horizontale Position, die immer relativ zum übergeordneten Element angegeben wird (optional, Standardwert ist 0);

`y:Number:` Vertikale Position, die immer relativ zum übergeordneten Element angegeben wird (optional, Standardwert ist 0);

`width:Number:` Breite (optional, andernfalls wird die Breite automatisch berechnet);

`height:Number:` Höhe (optional, andernfalls wird die Breite automatisch berechnet);

`trueUpSource:String`: Pfadangabe für das Jpeg-Bild des selektierten Normalzustandes für ausgeschalteten Ton (muss angegeben werden);

`trueOverSource:String`: Pfadangabe für das Jpeg-Bild des selektierten Darüberzustandes für ausgeschalteten Ton (optional, andernfalls wird der Normalzustand angezeigt);

`trueDownSource:String`: Pfadangabe für das Jpeg-Bild des selektierten Gedrücktzustandes für ausgeschalteten Ton (optional, andernfalls wird der Normalzustand angezeigt);

`falseUpSource:String`: Pfadangabe für das Jpeg-Bild des unselektierten Normalzustandes für eingeschalteten Ton (muss angegeben werden);

`falseOverSource:String`: Pfadangabe für das Jpeg-Bild des unselektierten Darüberzustandes für eingeschalteten Ton (optional, andernfalls wird der Normalzustand angezeigt);

`falseDownSource:String`: Pfadangabe für das Jpeg-Bild des unselektierten Gedrücktzustandes für eingeschalteten Ton (optional, andernfalls wird der Normalzustand angezeigt);

2.3.9.2. Beispiel

```
<UIMuteButton id="masterAndSlaveBtn" muteMode="masterAndSlaveMute" depth="17"
x="360" y="100" width="23" height ="23" falseUpSource="layout/muteOf.jpg"
falseOverSource="layout/muteOver.jpg" trueUpSource="layout/muteOn.jpg"
trueOverSource="layout/muteOf.jpg"/>
```

2.3.10. UIPlayhead

UIPlayhead kommt als Abspielkopf nur als untergeordnete Komponente der UITimeLine zum Einsatz. Dort dient diese Komponente als Bedienelement, um die Position in einem Videoclip anzuzeigen und zu verändern.

2.3.10.1. Parameter

`id:String`: Instanzname in Flash (muss angegeben werden);

`depth:Number`: Darstellungstiefe in der Stapelreihenfolge (muss angegeben werden);

`width:Number:` Breite (optional, andernfalls wird die Breite automatisch berechnet);

`height:Number:` Höhe (optional, andernfalls wird die Breite automatisch berechnet);

`upSource:String`: Pfadangabe für das Jpeg-Bild des Normalzustandes (muss angegeben werden);

`overSource:String`: Pfadangabe für das Jpeg-Bild des Darüberzustandes (optional, andernfalls wird der Normalzustand angezeigt);

`downSource:String`: Pfadangabe für das Jpeg-Bild des Gedrücktzustandes (optional, andernfalls wird der Normalzustand angezeigt);

`xMin:Number`: Minimale X-Position (muss angegeben werden);

`xMax:Number`: Maximale X-Position (muss angegeben werden);

`y:Number`: Y-Position (muss angegeben werden);

xMin und xMax definieren den «von-bis»-Bereich innerhalb der Timeline, in welchem sich der Playhead bewegen kann.

2.3.10.2. Beispiel

```
<UIPlayhead xMin="123" xMax="123" depth="2" id="downButton" x="0" y="174"
width="20" height="20" upSource="up.jpg" overSource="over.jpg"/>
```

2.3.11. UIPosition

Die Komponente UIPosition zeigt die x- und y-Koordinaten des Mauszeigers innerhalb des UIMediaDisplays (id="master"). Diese Komponente wird in der Regel nur während der Erstellung einer Anwendung aktiviert, um die Position und Größe der zu erstellenden Hotspots bestimmen zu können.

2.3.11.1. Parameter

`id:String`: Instanzname in Flash (muss angegeben werden);

`depth:Number`: Darstellungstiefe in der Stapelreihenfolge (muss angegeben werden);

`x:Number:` Horizontale Position, die immer relativ zum übergeordneten Element angegeben wird (optional, Standardwert ist 0);

`y:Number:` Vertikale Position, die immer relativ zum übergeordneten Element angegeben wird (optional, Standardwert ist 0);

`width:Number:` Breite (optional, andernfalls wird die Breite automatisch berechnet);

`height:Number:` Höhe (optional, andernfalls wird die Breite automatisch berechnet);

`font:String`: Optionale Angabe der Schrift oder der Schriften durch Kommata getrennt (Standard ist Verdana);

`fontColor:Number`: Optionale Angabe der Textfarbe (Standard ist 0x000000 für Schwarz);

`backgroundColor:Number`: Optionale Angabe der Hintergrundfarbe (ist eine Hintergrundgrafik eingebunden, dann ist dieser Parameter ohne Wirkung);

`bold:Boolean`: Optionale Angabe des Schriftstils Fett (Standard ist false);

`italic:Boolean`: Optionale Angabe des Schriftstils Kursiv (Standard ist false);

`underline:Boolean`: Optionale Angabe des Schriftstil Unterstrichen (Standard ist false);

Eine optionale Hintergrundgrafik wird als untergeordnete UIImage-Komponente mit der id „background“ angegeben;

2.3.11.2. Beispiel

```
<UIPosition depth="6" id="position" x="405" y="100" width="353" height="200" />
```

2.3.12. UIProgressBar

Diese Komponente wird zur Darstellung des Prozessbalkens (Ladefortschritt) in der Komponente UITimeLine eingesetzt.

2.3.12.1. Parameter

`id:String`: Instanzname in Flash (muss als „progressBar“ angegeben werden);

`depth:Number`: Darstellungstiefe in der Stapelreihenfolge (muss angegeben werden);

`x:Number:` Horizontale Position, die immer relativ zum übergeordneten Element angegeben wird (optional, Standardwert ist 0);

`y:Number:` Vertikale Position, die immer relativ zum übergeordneten Element angegeben wird (optional, Standardwert ist 0);

`width:Number:` Breite (optional, andernfalls wird die Breite automatisch berechnet);

`height:Number:` Höhe (optional, andernfalls wird die Höhe automatisch berechnet);

`source:String`: Pfadangabe für das Jpeg-Bild (muss angegeben werden);

2.3.12.2. Beispiel

```
<UIProgressBar id="progressBar" depth="1" x="7" y="5" width="300" height="6"
source="layout/playheadprogress.jpg"/>
```

2.3.13. UIScrollBar

Rollbalken werden (optional) als UIScrollBar-Komponente innerhalb der UITextArea- und UIList-Komponenten genutzt und positionieren sich automatisch rechts neben diesen Komponenten.

2.3.13.1. Parameter

`id:String`: Instanzname in Flash (muss als „scrollBar“ angegeben werden);

`depth:Number`: Darstellungstiefe in der Stapelreihenfolge (muss angegeben werden);

Der Hintergrund des Rollbalkens wird über die UIImage-Komponente mit der id „scrollTrack“ angegeben;

Eine Schaltfläche für den Rollbalken muss mit der id „scrollThumb“ als UIButton-Komponente angegeben werden;

Eine optionale Schaltfläche zum nach oben Navigieren wird als untergeordnete UIButton-Komponente mit der id „scrollUpArrow“ angegeben – wenn diese Schaltfläche gewünscht ist muss auch die Runter-Schaltfläche (scrollDownArrow) initialisiert werden;

Eine optionale Schaltfläche zum nach unten Navigieren wird als untergeordnete UIButton-Komponente mit der id „scrollDownArrow“ angegeben – wenn diese Schaltfläche gewünscht ist muss auch die Rauf-Schaltfläche (scrollUpArrow) initialisiert werden;

2.3.13.2. Beispiel

```
<UIScrollBar depth="1" id="scroller">
   <UIImage id="scrollTrack" depth="0" source="layout/menusliderbox.jpg" />
   <UIButton depth="1" id="scrollUpArrow" x="2" upSource="layout/scroll/scroll-
up.jpg"/>
   <UIButton depth="3" id="scrollThumb" x="2" upSource="layout/scroll/scroll-
thumb.jpg"/>
   <UIButton depth="2" id="scrollDownArrow" x="2" upSource="layout/scroll/scroll-
down.jpg"/>
</UIScrollBar>
```

2.3.14. UITaskBar

Dies ist die Seitenleiste, welche am oberen Rand erscheint, wenn man mit der Maus an den oberen Rand der MovieBench fährt. Die Komponente enthält links die «Produktinfo» und rechts zwei Button «Help» und «Quit».
Diese Komponente ist automatisch immer in der Anwendung enthalten und kann nicht durch das Weglassen in der Layout-Beschreibung entfernt werden. Es können mehrere Designvarianten der UITaskBar zur Verfügung stehen, aus denen der Autor eine mittels des Attributes type wählen kann.

2.3.14.1. Parameter

`id:String`: Instanzname in Flash (optional);

`type:Number:` Optionale Angabe der Designvariante anhand einer Identifikationsnummer (Standard ist 1);

`helpUrl:String:` Angabe des Hilfe-Links, also der Webseite, wo Sie Ihren BenutzerInnen Hilfe zu diesem Lernmodul anbieten.

`quitUrl:String:` Angabe des Beenden-Links, also der Webseite, wo Sie Ihre BenutzerInnen hinführen wollen, wenn jene dieses Lernmodul beenden.

2.3.14.2. Beispiel

```
<UITaskBar id="myTaskBar" type="2"
helpUrl="http://myserver.ch/mypath/myhelp.html"
quitUrl="http:// myserver.ch/mypath/myquit.html" />
```

2.3.15. UITextArea

Die UITextArea wird zur Anzeige von Text verwendet. Es können mehrere UITextArea platziert werden, wobei jede eine unique id benötigt.
Spezialfall: der Inhalt derjenigen UITextArea mit der id «textArea» kann während des Abspielens eines Clips mit Hilfe der Aktion «ShowTextAction» dynamisch verändert werden.

2.3.15.1. Parameter

`id:String`: Instanzname in Flash (muss angegeben werden);

`depth:Number`: Darstellungstiefe in der Stapelreihenfolge (muss angegeben werden);

`x:Number:` Horizontale Position, die immer relativ zum übergeordneten Element angegeben wird (optional, Standardwert ist 0);

`y:Number:` Vertikale Position, die immer relativ zum übergeordneten Element angegeben wird (optional, Standardwert ist 0);

`width:Number:` Breite (optional, andernfalls wird die Breite automatisch berechnet);

`height:Number:` Höhe (optional, andernfalls wird die Breite automatisch berechnet);

Eine optionale Hintergrundgrafik wird als untergeordnete UIImage-Komponente mit der id „background“ angegeben;

Eine optionaler Rollbalken wird als untergeordnete UIScrollBar-Komponente mit der id „scrollBar“ angegeben;

2.3.15.2. Beispiel

```
<UITextArea depth="6" id="textArea" x="405" y="100" width="353"
height="200" />
```

Hinweis

Besonderheiten

Diese Komponente erlaubt zusätzlich auch alle Eigenschaften der Flash Klasse TextField. Um die Hintergrundfarbe definieren zu können, muss wie in Flash zuvor die Eigenschaft background auf true gesetzt werden und anschließend die Hintergrundfarbe über backgroundColor eingestellt werden. Falls die Breite und Höhe nicht definiert sind, wird der Text automatisch nach links laufen (`autosize="left"`) - außer diese Eigenschaft ist explizit deaktiviert bzw. ist nicht angegeben worden. Siehe dazu das Kapitel «Tipps».

2.3.16. UITimeLine

Die Komponente UITimeLine wird verwendet, um den zeitlichen Ablauf des aktuellen Videos zu zeigen und zu steuern. Mit Hilfe der UIProgressBar wird innerhalb der UITimeLine der Ladestatus dargestellt. UIPlayhead erlaubt es, den zeitlichen Verlauf des Videoclips abzulesen und zu verändern (also zu einer beliebigen bereits geladenen Position zu wechseln). Bitte beachten Sie, dass dabei auf die Existenz von Hotspots getestet wird.

2.3.16.1. Parameter

`id:String`: Instanzname in Flash (muss angegeben werden);

`depth:Number`: Darstellungstiefe in der Stapelreihenfolge (muss angegeben werden);

`x:Number:` Horizontale Position, die immer relativ zum übergeordneten Element angegeben wird (optional, Standardwert ist 0);

`y:Number:` Vertikale Position, die immer relativ zum übergeordneten Element angegeben wird (optional, Standardwert ist 0);

`width:Number:` Breite (optional, andernfalls wird die Breite automatisch berechnet);

`height:Number:` Höhe (optional, andernfalls wird die Breite automatisch berechnet);

Der Abspielkopf wird als untergeordnete UIPlayhead-Komponente mit der id „playhead“ angegeben;

Der Fortschrittsbalken wird als untergeordnete UIProgressBar-Komponente mit der id „progressBar“ angegeben;

Eine optionale Hintergrundgrafik wird als untergeordnete UIImage-Komponente mit der id „background“ angegeben;

2.3.16.2. Beispiel

```
<UITimeLine depth="4" id="timeline" x="30" y="355" width="353" height="20">
   <UIImage id="background" depth="0" x="0" y="0" source="layout/playheadbox.jpg"/>
   <UIProgressBar id="progressBar" depth="1" x="7" y="5"
source="layout/playheadprogress.jpg"/>
   <UIPlayhead depth="2" id="playhead" xMin="7" xMax="346" y="1"
upSource="layout/playhead.jpg" overSource="layout/playhead.jpg"
downSource="layout/playhead.jpg" />
</UITimeLine>
```

2.3.17. UITitle

Die Komponente UITitle zeigt den Titel (`<ClipLabel>`) des aktuellen Videoclips an.

2.3.17.1. Parameter

`id:String`: Instanzname in Flash (muss angegeben werden);

`depth:Number`: Darstellungstiefe in der Stapelreihenfolge (muss angegeben werden);

`x:Number:` Horizontale Position, die immer relativ zum übergeordneten Element angegeben wird (optional, Standardwert ist 0);

`y:Number:` Vertikale Position, die immer relativ zum übergeordneten Element angegeben wird (optional, Standardwert ist 0);

`width:Number:` Breite (optional, andernfalls wird die Breite automatisch berechnet);

`height:Number:` Höhe (optional, andernfalls wird die Breite automatisch berechnet);

`font:String`: Optionale Angabe der Schrift oder der Schriften durch Kommata getrennt (Standard ist „Verdana“);

`fontColor:Number`: Optionale Angabe der Textfarbe (Standard ist 0x000000 für Schwarz);

`backgroundColor:Number`: Optionale Angabe der Hintergrundfarbe (ist eine Hintergrundgrafik eingebunden, dann ist dieser Parameter ohne Wirkung);

`bold:Boolean`: Optionale Angabe des Schriftstils Fett (Standard ist false);

`italic:Boolean`: Optionale Angabe des Schriftstils Kursiv (Standard ist false);

`underline:Boolean`: Optionale Angabe des Schriftstil Unterstrichen (Standard ist false);

Eine optionale Hintergrundgrafik wird als untergeordnete UIImage-Komponente mit der id „background“ angegeben;

2.3.17.2. Beispiel

```
<UITitle id="titelbox" depth="12" x="30" y="65" width="353" height="25"
font="Arial, Helvetica" fontSize="14" fontColor="0" backgroundColor="0xCCCCCC"
bold="true">
<UIImage id="background" depth="1" source="layout/box.jpg"/>
</UITitle>
```

2.3.18. UITotalTime

Diese Komponente zeigt die Dauer aller Videoclips in der Form hh:mm:ss (Stunden, Minuten, Sekunden). Diese errechnet sich aus der Summe aller in der Playlist angegebenen Videoclips. Dabei handelt es sich lediglich um einen Richtwert.

2.3.18.1. Parameter

`id:String`: Instanzname in Flash (muss angegeben werden);

`depth:Number`: Darstellungstiefe in der Stapelreihenfolge (muss angegeben werden);

`x:Number:` Horizontale Position, die immer relativ zum übergeordneten Element angegeben wird (optional, Standardwert ist 0);

`y:Number:` Vertikale Position, die immer relativ zum übergeordneten Element angegeben wird (optional, Standardwert ist 0);

`width:Number:` Breite (optional, andernfalls wird die Breite automatisch berechnet);

`height:Number:` Höhe (optional, andernfalls wird die Breite automatisch berechnet);

`font:String`: Optionale Angabe der Schrift oder der Schriften durch Kommata getrennt (Standard ist Arial);

`fontColor:Number`: Optionale Angabe der Textfarbe (Standard ist 0x000000 für Schwarz);

`backgroundColor:Number`: Optionale Angabe der Hintergrundfarbe (ist eine Hintergrundgrafik eingebunden, dann ist dieser Parameter ohne Wirkung);

`bold:Boolean`: Optionale Angabe des Schriftstils Fett (Standard ist false);

`italic:Boolean`: Optionale Angabe des Schriftstils Kursiv (Standard ist false);

`underline:Boolean`: Optionale Angabe des Schriftstil Unterstrichen (Standard ist false);

Eine optionale Hintergrundgrafik wird als untergeordnete UIImage-Komponente mit der id „background“ angegeben;

2.3.18.2. Beispiel

```
<UITotalTime depth="6" id="totalTime" x="405" y="100" width="353" height="200" />
```

3. Playlist

3.1. Übersicht und Struktur

Die XML-Datei **playlist.xml** ist hierarchisch gegliedert und besteht im Wesentlichen aus folgenden Elementen:

```
<?xml version="1.0" encoding="utf-8"?>          XML-Header
<Playlist attributes >                          äusserstes Container-Element
   <Clip attributes>                            erster Videoclip
      <ClipLabel> ... </ClipLabel>              Label; wird in UITitle dargestellt
      <ClipMenuLabel> ... </ClipMenuLabel>      wird als Item in UIList dargestellt
      <HotspotList>                             optionale Liste mit Hotspots
         <RectHotspot attributes >              Rechteckiger Hotspot
            <ActionList>                        Liste von Aktionen dieses Hotspots
               ...                              einzelne Aktionen

            </ActionList>
         </RectHotspot>
         weitere Hotspots ...                   weitere Hotspots (Ellipsen, Polygon..)
      </HotspotList>
   </Clip>
   weitere Clips ...                            beliebige Anzahl weiterer Clips
</Playlist>
```

Eine Playlist besteht aus einem Header und einem äussersten Container `<Playlist .. >`. Innerhalb der Playlist werden alle abzuspielenden Videoclips `<Clip .. >` in der zu erscheinenden Reihenfolge aufgelistet. Pro Clip können `<ClipLabel>`, `<ClipMenuLabel>` und eine optionale Liste mit Hotspots `<HotspotList>` definiert werden.

3.2. Elemente der Playlist

3.2.1. Das Element <Playlist .. >

Das Element <Playlist .. > enthält folgende Parameter:

`autostart:Boolean`: Dieser optionale Wert beschreibt, ob der erste Clip automatisch gestartet werden soll (Standard ist true);

`proceedMode:String`: Optionale Angabe für das Fortsetzen von nachfolgenden Clips in der Form „`stop`“ (stoppt beim Erreichen des Endes eines Clips), „`next`“ (geht zum nächsten Clip und stoppt), „`continuous`“ (setze des Abspielen mit dem nächsten Clip fort) oder „`repeat`“ (wiederhole den aktuellen Clip) (Standard ist „`continuous`“)

3.2.2. Das Element <Clip .. >

Das Element <Clip .. > enthält folgende Parameter:

`master:String`: URL zum primären Videoclip (muss angegeben werden);

`slave:String`: URL zum sekundären Videoclip (optional)

`duration:Number`: Dauer in Millisekunden (Ticks) (muss angegeben werden);

`id:String`: Bezeichner für den Clip (optional, wird jedoch bei Verwendung der gotoClip-Aktion als Sprungmarke benötigt);

`fps:Number`: Bildrate in Bildern pro Sekunde (muss angegeben werden);

`bufferTime:Number`: Größe des Puffers zum Vorausladen in Millisekunden (optional, Standard ist 5000);

`steppingSpeed:Number`: Schrittweite beim schrittweise Vor- und Zurückgehen in Millisekunden (optional, Standard ist 200);

`windingSpeed:Number`: Schrittweite beim Vor- und Zurückspulen in Millisekunden (optional, Standard ist 3000);

`proceedMode:String`: Optionale Angabe für das Fortsetzen von nachfolgenden Clips in der Form „`stop`“ (stoppt beim Erreichen des Endes eines Clips), „`next`“ (geht zum nächsten Clip und stoppt), „`continuous`“ (setze des Abspielen mit dem nächsten Clip fort) oder „`repeat`“ (wiederhole den aktuellen Clip) (Standard ist „`inherit`“, was für die Einstellung des übergeordneten proceedMode auf PlayList-Ebene steht);

Die Bezeichnung des Clips für die Anzeige im Titel (UITitel) wird als untergeordnetes ClipLabel-Element angegeben.

Der Bezeichner für die Anzeige im Menü (UIList) über das untergeordnete ClipMenuLabel-Element angegeben.

Hinweis

Probleme bei der Angabe von Schrittweiten

Aufgrund der gewählten Kodierung eines Videos kann es bei sehr kleinen Schrittweiten (steppingSpeed) zu Problemen kommen. Dies hängt u. a. mit der Anzahl der Schlüsselbilder (keyframes) zusammen. Je dichter diese liegen, desto kleiner können die Schrittweiten gewählt werden. Generell kann man sagen, dass die kleinste Schrittweite grösser sein muss als der Abstand zwischen zwei Schlüsselbildern.

3.2.2.1. Beispiel

```
<?xml version="1.0" encoding="utf-8"?>
<Playlist autostart="true" proceedMode="continuous" >
   <Clip master="videos/C3.1.11.flv" proceedMode="inherit" duration="45000"
         id="clip1" fps="15" bufferTime="5" steppingSpeed="300"
         windingSpeed="5000">
      <ClipLabel>Einführung in die Legasthenie</ClipLabel>
      <ClipMenuLabel>Einführung</ClipMenuLabel>
      <HotspotList>
         <RectHotspot startTime="10000" stopTime="15000"
                      color="0xff0000" alpha="75" width="100" height="100">
            <ActionList>
               <StopAction event="onShow"/>
               <ShowTextAction event="onShow"><![CDATA[<html>klicke auf das
                     Viereck um den Film fortzusetzen</html>]]></ShowTextAction>
               <ShowTextAction event="onClick"><![CDATA[<html>DANKE
                     ;)</html>]]></ShowTextAction>
               <ContinueAction event="onClick" />
            </ActionList>
         </RectHotspot>
      </HotspotList>
   </Clip>
   <Clip master="videos/C3.1.2.flv" slave="/videos/C3.1.2b.flv"
         proceedMode="0" duration="19" id="clip2" fps="15">
      <ClipLabel>Vergleich von Recht- mit Link-Schreibung</ClipLabel>
```

```
      <ClipMenuLabel>Vergleich</ClipMenuLabel>
   </Clip>
</Playlist>
```

3.2.3. Hotspots

Hotspots sind Interaktionselemente, die im Ablauf eines Videoclips erscheinen und wieder verschwinden können. Dabei sind auch mehrere parallele Hotspots erlaubt. Im einfachsten Fall führt ein Hotspot zu der Anzeige eines Textes in der TextArea mit der id „textArea“. Zusätzlich darf ein Hotspot innerhalb der UIMediaDisplay-Komponente mit der id „master“ auch visualisiert werden. Hotspots erlauben die Ausführung von **Actions**, die in Folge eines **Events** ausgelöst werden.
Das folgende Beispiel zeigt einen Hotspot vom Typ `BlindHotspot`, welcher zum Zeitpunkt t=3.5sec (`startTime="3500"`, `event="onShow"`) dafür sorgt, dass in der TextArea mit der id „textArea“ ein Text erscheint (`ShowTextAction`). Am Ende des Hotspots bei t=7.4sec (`stopTime="7400"`, `event="onHide"`) wird ein neuer Text angezeigt. Ein `BlindHotspot` ist die einfachste Form eines Hotspots. Dieser zeigt lediglich Text an und erlaubt keine Interaktion.

```
<HotspotList>
   <BlindHotspot startTime="3500" stopTime="7400">
      <ActionList>
         <ShowTextAction event="onShow">
            <![CDATA[<html><B>Unsichtbarer </B><br><i>Hotspot </i> da</html>]]>
         </ShowTextAction>
         <ShowTextAction event="onHide">
            <![CDATA[<html>Unsichtbarere Hotspot ist weg</html>]]>
         </ShowTextAction>
      </ActionList>
   </BlindHotspot>
</HotspotList>
```

MovieBench kennt vier Arten von Hotspots – BlindHotspot, RectHotspot, EllipsisHotspot und PolygonHotspot.

3.2.3.1. BlindHotspot

Ein unsichtbarer Hotspot dient zur automatischen Ausführung von Aktionen zu einem fest definierten Zeitpunkt.

3.2.3.1.1. Parameter

`startTime:Number`: Zeitpunkt (in Millisekunden) der Aktivierung (muss angegeben werden);
`stopTime:Number`: Zeitpunkt (in Millisekunden) der Deaktivierung (optional);
Als untergeordnetes Element wird eine Liste mit den Aktionen angegeben;

3.2.3.1.2. Beispiel

```
<BlindHotspot startTime="3000" stopTime="8000">
   <ActionList>
      <!-- Hier stehen die Aktionen -->
   </ActionList>
</BlindHotspot>
```

3.2.3.2. RectHotspot

Dieser Hotspot wird als Rechteck angezeigt. Zugeordnete Aktionen können automatisch oder durch den Anwender ausgelöst werden.

3.2.3.2.1. Parameter

`startTime:Number`: Zeitpunkt (in Millisekunden) der Aktivierung (muss angegeben werden);

`stopTime:Number`: Zeitpunkt (in Millisekunden) der Deaktivierung (optional);

`color`:Number: Farbe des Hotspots (optional, Standard ist 0xFF0000 für rot);

`alpha:Number`: Transparenz von 0 für durchsichtig bis 100 für deckend (optional, der Standard ist 100);

`width:Number`: Breite (optional);

`height:Number`: Höhe (optional);

`x:Number`: Horizontale Position (optional);

`y:Number`: Vertikale Position (optional);

`borderAlpha:Number`: Transparenz des Randes (optional, der Standard ist 0);

`borderSize:Number`: Dicke des Randes (optional, der Standard ist keine Angabe, d. h. kein Rahmen);

`borderColor:Number`: Farbe des Randes (optional, der Standard ist 0x000000);

Als untergeordnetes Element wird eine Liste mit den Aktionen angegeben;

3.2.3.2.2. Beispiel

```
<RectHotspot startTime="10000" stopTime="15000"
             color="0xFF0000" alpha="75" width="100" height="100">
   <ActionList>
      <!-- Hier stehen die Aktionen -->
   </ActionList>
</RectHotspot>
```

3.2.3.3. EllipsisHotspot

Dieser Hotspot wird als Oval angezeigt. Zugeordnete Aktionen können automatisch oder durch den Anwender ausgelöst werden.

3.2.3.3.1. Parameter

`startTime:Number`: Zeitpunkt (in Millisekunden) der Aktivierung (muss angegeben werden);

`stopTime:Number`: Zeitpunkt (in Millisekunden) der Deaktivierung (optional);

`color`:Number: Farbe des Hotspots (optional, der Standard ist 0xFF0000 für rot);

`alpha:Number`: Transparenz von 0 für durchsichtig bis 100 für deckend (optional, der Standard ist 100);

`width:Number`: Breite (optional);

`height:Number`: Höhe (optional);

`x:Number`: Horizontale Position (optional);

`y:Number`: Vertikale Position (optional);

`borderAlpha:Number`: Transparenz des Randes (optional, der Standard ist 0);

`borderSize:Number`: Dicke des Randes (optional, der Standard ist keine Angabe, d. h. kein Rahmen);

`borderColor:Number`: Farbe des Randes (optional, der Standard ist 0x000000);

Als untergeordnetes Element wird eine Liste mit den Aktionen angegeben;

3.2.3.3.2. Beispiel

```
<EllipsisHotspot startTime="10000" stopTime="15000"
                 color="0xFF00cc" alpha="75"
                 width="100" height="100">
   <ActionList>
      <!-- Hier stehen die Aktionen -->
   </ActionList>
</EllipsisHotspot>
```

3.2.3.4. PolygonHotspot

Dieser Hotspot wird als Vieleck angezeigt. Zugeordnete Aktionen können automatisch oder durch den Anwender ausgelöst werden.

3.2.3.4.1. Parameter

`startTime:Number`: Zeitpunkt (in Millisekunden) der Aktivierung (muss angegeben werden);

`stopTime:Number`: Zeitpunkt (in Millisekunden) der Deaktivierung (optional);

`color`:Number: Farbe des Hotspots (optional, Standard ist 0xFF0000 für rot);

`alpha:Number`: Transparenz von 0 für durchsichtig bis 100 für deckend (optional, der Standard ist 100);

`width:Number`: Breite (optional);

`height:Number`: Höhe (optional);

`x:Number`: Horizontale Position (optional);

`y:Number`: Vertikale Position (optional);

`borderAlpha:Number`: Transparenz des Randes (optional, der Standard ist 0);

`borderSize:Number`: Dicke des Randes (optional, der Standard ist keine Angabe, d. h. kein Rahmen);

`borderColor:Number`: Farbe des Randes (optional, der Standard ist 0x000000);

Als untergeordnetes Element wird eine Liste mit den Aktionen angegeben;

Die Form wird durch eine untergeordnete Liste von Ankerpunkten beschrieben (siehe Beispiel);

3.2.3.4.2. Beispiel

```
<PolygonHotspot startTime="10000" stopTime="15000"
                color="0x00ccFF" alpha="75"
                width="100" height="100">
   <ActionList>
      <!-- Hier stehen die Aktionen -->
   </ActionList>
   <AnchorPointList>
      <AnchorPoint x="50" y="0" />
      <AnchorPoint x="62.5" y="37" />
      <AnchorPoint x="100" y="38" />
      <AnchorPoint x="70" y="62" />
   </AnchorPointList>
</PolygonHotspot>
```

3.2.4. Events

Drei mögliche Events können eine Aktion auslösen:

onShow – Zeitpunkt, an dem der Hotspot erscheint und aktiv wird.
onHide – Zeitpunkt, an dem der Hotspot wieder verschwindet.
onClick – falls die Benutzerin auf den Hotspot klickt.

3.2.5. Actions

Jeder Hotspot kann eine Liste von Aktionen enthalten, die entweder automatisch ausgeführt oder durch den Benutzer ausgelöst werden. Zum Beispiel kann ein Videoclip beim Erreichen eines Hotspots angehalten werden oder weiterlaufen. Und wird der Hotspot angeklickt, dann kann (sowohl als auch) ...
... ein neuer Text in die Textbox eingeblendet werden (`ShowTextAction`).
... der Clip gesteuert werden – anhalten oder fortsetzen (`StopAction`, `ContinueAction`).
... eine externe Webseite geöffnet werden (`GetUrlAction`).
... zu einem anderen Videoclip gesprungen werden (`GotoClip`).

Der Zeitpunkt, zu dem eine Aktion ausgelöst wird, hängt vom zugeordneten Event ab. Alle fünf Actions können mit jedem der drei Events kombiniert werden.

3.2.5.1. ShowTextAction

Diese Aktion zeigt einen Text in derjenigen Komponente UITextArea an, welche mit der id „textArea“ versehenen ist.

3.2.5.1.1. Parameter

`event:String`: Angabe des Ereignisses, dass die Aktion auslöst (muss angegeben werden);
Als untergeordnetes Element wird der anzuzeigende Text festgelegt;

3.2.5.1.2. Beispiel

```
<ShowTextAction event="onHide">
  <![CDATA[<html>Unsichtbarer Hotspot ist weg</html>]]>
</ShowTextAction>
```

3.2.5.2. StopAction

Diese Aktion stoppt den aktuell laufenden Videoclip (oder beide, falls «master» und «slave» vorhanden).

3.2.5.2.1. Parameter

`event:String`: Angabe des Ereignisses, dass die Aktion auslöst (muss angegeben werden);

3.2.5.2.2. Beispiel

```
<StopAction event="onShow"/>
```

3.2.5.3. ContinueAction

Diese Aktion setzt das Abspielen des aktuellen Videoclips fort.

3.2.5.3.1. Parameter

`event:String`: Angabe des Ereignisses, dass die Aktion auslöst (muss angegeben werden);

3.2.5.3.2. Beispiel

```
<ContinueAction event="onClick" />
```

3.2.5.4. GetUrlAction

Diese Aktion öffnet in einem Browserfenster die gewünschte Webseite.

3.2.5.4.1. Parameter

`event:String`: Angabe des Ereignisses, dass die Aktion auslöst (muss angegeben werden);

`url:String`: Angabe der URL;

`target:String`: Optionale Angabe des Ziels (z. B. der Name eines Fensters oder Frames);

3.2.5.4.2. Beispiel

```
<GetUrlAction event="onClick" url="http://www.unibas.ch/patho/" target="_blank" />
```

3.2.5.5. GotoClipAction

Diese Aktion verzweigt zu einer anderen Position in der Playlist.

3.2.5.5.1. Parameter

`event:String`: Angabe des Ereignisses, dass die Aktion auslöst (muss angegeben werden);

`clipId:String`: Angabe der Zielmarke als id eines Clips (muss angegeben werden);

`position:Number`: Angabe der Position innerhalb des Clips in Millisekunden (optional);

3.2.5.5.2. Beispiel

```
<GotoClipAction event="onClick" clipId="clip9" position="30000" />
```

4. Tipps

Da sich die Angabe von Text im HTML-Format nicht immer leicht gestaltet, haben wir hier die wichtigsten Informationen zum Thema für Sie zusammengestellt. Ebenfalls finden Sie in diesem Kapitel Informationen zu dem Basisverzeichnis einer Flash-Anwendung.

4.1. HTML

Seit Flash 5 gibt es die Unterstützung von HTML-Formaten und Hyperlinks (Querverweise) innerhalb eines Textblockes. Die Anzeige eines HTML-Textes inklusive der Formatierung war bis dahin innerhalb Flash nicht möglich. Die **Texteingabe formatieren** Sie, indem Sie die folgenden HTML-Tags verwenden:

`<A>`
`<B>`
`<FONT COLOR>`
`<FONT FACE>`
`<FONT SIZE>`
`<I>`
`<P>`
`<U>`
`<BR>` bzw. `<BR />` (der Zeilenumbruch ist zwar nicht dokumentiert, funktioniert aber)

Neu in Flash MX hinzugekommen sind die HTML-Tags `<LI>` und `<Textformat>`. Mit `<LI>` realisieren Sie Aufzählungen, wobei diese nur als **Punktaufzählung** dargestellt werden. `<Textformat>` unterstützt Sie bei der Formatierung Ihres Textes. Mögliche Parameter des Tags sind:

`<TEXTFORMAT LEFTMARGIN="n"`
Eigenschaft, mit der man den linken Rand eines Absatzes definiert (n = ganzzahliger Wert für den Abstand in Punkt).

`<TEXTFORMAT RIGHTMARGIN="n"`
Eigenschaft, mit der man den rechten Rand eines Absatzes definiert (n = ganzzahliger Wert für den Abstand in Punkt).

`<TEXTFORMAT BLOCKINDENT="n"`
Definition der Einrückung eines Textblockes in Punkt (n = ganzzahliger Wert).

`<TEXTFORMAT INDENT="n"`
Mit dieser Eigenschaft bestimmen Sie den Abstand vom linken Rand bis zum ersten Zeichen eines Absatzes. Folgezeilen des Absatzes werden nicht eingerückt (n = ganzzahliger Wert).

`<TEXTFORMAT LEADING="n"`
Einstellen des Zeilenabstandes (n = ganzzahliger Wert).

`<TEXTFORMAT TABSTOPPS="n1,n2,...,nN"`
Geben Sie die jeweiligen Werte der gewünschten Tabstopps als Parameter für diese Eigenschaft ein. Die Werte (n1 bis nN) müssen positiv ganzzahlig sein.

Seit Flash MX 2004 ist erstmalig auch die Angabe von Bild-Tags (**Image-Tags**) erlaubt: `<img src='datei'>`. Flash versteht als Medien MovieClips (*.swf), Jpeg-Bilder (*.jpg und *.jpeg, der Flash Player unterstützt jedoch keine progressiven JPEG-Dateien) sowie in einer Anwendung eingebettete Symbole (Verknüpfungsbezeichner). Des Weiteren sind die folgenden Attribute zulässig:

`<img src='datei' height=100 width=100>`: Optional geben Sie die Breite und Höhe eines Elementes in Pixeln vor. Andernfalls stellt Flash das Element in der Originalgröße dar.

`<img src='datei' align='ausrichtung'>`: Optional geben Sie einen horizontale Ausrichtung an. Gültige Werte für die Ausrichtung sind `left` (links) und `right` (rechts). Der Standardwert lautet `left`.

`<img src='datei' hspace=8 vspace=8>`: Optional legen Sie einen horizontalen (`hspace`) bzw. vertikalen (`vspace`) Bereich um das Element herum an, in dem kein Text zu sehen ist. Der Standardwert lautet in beiden Fällen 8.

Dieser Abschnitt ist dem Buch «Flash MX 2004» von Sascha Wolter entnommen (siehe Kapitel «Bücher»).

4.2. Basisverzeichnis

Normalerweise zeigt ein Webbrowser Flash Player-Filme in ein HTML-Dokument eingebettet an. Um den Flash Player-Film in einer HTML-Seite darzustellen, gibt es zwei verschiedene Varianten. Für den Internet Explorer unter Windows wird der Film mit dem OBJECT-Tag als Objekt integriert, andere Webbrowser (insbesondere die von Netscape) betten Filme über EMBED ein. Die unterschiedlichen Strategien von Microsoft und Netscape machen nicht nur das Einbetten eines Flash Player-Films in HTML-Seiten umständlich, auch die Steuerung über JavaScript wird in beiden „Welten“ unterschiedlich gehandhabt.
Alle Parameter für EMBED werden innerhalb des öffnenden Tags angegeben:

```
<EMBED
 src="meinfilm.swf"
 WIDTH="550"
 HEIGHT="400"
 TYPE="application/x-shockwave-flash"
 PLUGINSPAGE="http://www.macromedia.com/go/getflashplayer">
</EMBED>
```

Beim OBJECT-Tag werden die Parameter HEIGHT (Höhe), WIDTH (Breite), CLASSID (Identifizierung der ActiveX-Steuerung) und CODEBASE (Position der ActiveX-Steuerung für den Download) innerhalb des öffnenden Tags festgelegt. Alle weiteren Parameter werden in speziellen PARAM-Tags festgehalten:

```
<OBJECT
 classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
 codebase=http://fpdownload.macromedia.com/pub/shockwave/cabs/flash/
          swflash.cab#version=7,0,0,0
 WIDTH="550"
 HEIGHT="400" >
 <PARAM NAME="movie" VALUE="meinfilm.swf">
</OBJECT>
```

Um in möglichst vielen Webbrowsern konsistente Ergebnisse zu erzielen, müssen immer **beide Methoden** verwendet werden. Um beide Tags gemeinsam zu nutzen, sollte das EMBED-Tag vor das schließende OBJECT-Tag platziert werden. Die jeweiligen Browser ignorieren die ihnen unbekannte Methode einfach:

```
<OBJECT
 classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
codebase=http://fpdownload.macromedia.com/pub/shockwave/cabs/flash/
         swflash.cab#version=7,0,0,0
 WIDTH="550"
 HEIGHT="400" >
 <PARAM NAME=movie VALUE="meinfilm.swf">
   <EMBED
    src="meinfilm.swf"
    WIDTH="550"
    HEIGHT="400"
    TYPE="application/x-shockwave-flash"
```

```
    PLUGINSPAGE="http://www.macromedia.com/go/getflashplayer">
  </EMBED>
</OBJECT>
```

Hinweis

Windows XP Service Pack 2

Im Rahmen des Service Pack 2 für Windows XP hat Microsoft das Verhalten des Betriebsystems und des Internet Explorers im Umgang mit ActiveX-Steuerungen wie dem Flash Player PlugIn verändert. Die Auswirkungen sind unter http://www.macromedia.com/support/service/servicepack2.html beschrieben. Der Parameter BASE (Basis) legt hier die Adresse eines Flash Player-Films für relative Pfade fest. Dieser Parameter ist optional und wird sowohl im OBJECT- als auch im EMBED-Tag verwendet. Er ist nützlich, wenn ein Flash-Film mit relativen Verweisen in ein anderes Verzeichnis platziert wird (Standard ist je nach Browser das Verzeichnis des Flash-Films oder der HTML-Seite).

4.3. Support

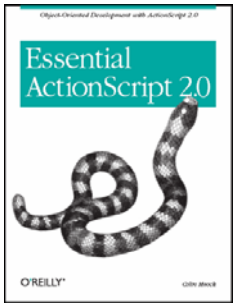
Unter http://moviebench.unibas.ch/support/ finden Sie Infos, die Ihnen bei der Erstellung eines Lermmoduls weiterhelfen.
Anschauungsbeispiele finden Sie unter http://moviebench.unibas.ch/demo/
Damit Ihnen der Einstieg leicher fällt, beginnen Sie am Besten mit unserem Beispiel «DesignSchreck», für welches Sie alle Dateien gezipped downloaden können unter http://moviebench.unibas.ch/start/

5. Bücher

Weitere Informationen zu der für MovieBench verwendeten Technologie Flash finden Sie in den folgenden Büchern. Diese behandeln auch die im Text angesprochenen Themen wie die Eigenschaften der TextField-Klasse und HTML 1.0 (u. a. für UILabel).
Insbesondere wurde das Kapitel Tipps dem ersten Buch entnommen. Sascha Wolter sei an dieser Stelle für sein freundliches Entgegenkommen gedankt!
Die zwei ersten Bücher sind «die» Standardwerke für Einstieg und die Vertiefung in Flash. Wer noch tiefer einsteigen will, benötigt in Ergänzung zu Sascha Wolters Büchern die zwei Bände von Colin Moock.

„Flash MX 2004 – Standard- und Professional-Version“
Autoren: Sascha Wolter, Marc Thiele
Preis: Euro 44,90
ISBN: 3898424685

„ActionScript 1 und 2 - Codedesign und Objektorientierung“
Autor: Sascha Wolter
Preis: Euro 44,90
ISBN: 3898422216
Erscheinungsdatum: Galileo-Press, 2004



„Essential ActionScript 2.0“
Colin Mock
Paperback: 528 pages
Publisher: O'Reilly; 1 edition (June 16, 2004)
ISBN: 0596006527

„ActionScript for Flash MX: The Definitive Guide, Second Edition“
Colin Moock
Paperback: 1104 pages
Publisher: O'Reilly; 2 edition (December 19, 2002)
ISBN: 059600396X