# 7 Programming with Video



Literature:


Clemens Szyperski: Component Software - Beyond Object-Oriented Programming. 2nd ed. Addison-Wesley 2002

M.D. McIlroy: Mass produced software components. In: Naur/Randell (eds.), Software Engineering, NATO Scientific Affairs Div. 1969 (http://www.cs.dartmouth.edu/~doug/components.txt)

# Software Components

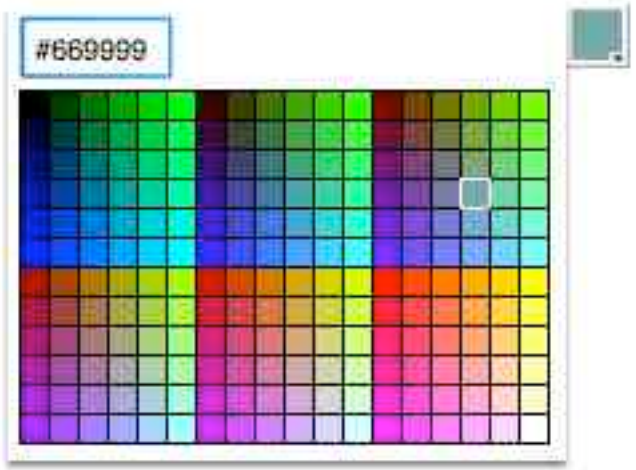
- *Software component:* "A ***software component*** is a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third parties."
  ECOOP 1996, Workshop on Component-oriented Programming
- Components for visual development environments:
  - Provide well-defined functionality
  - Can be dragged from palette to working area (creating an instance)
  - Can be adjusted by setting parameter values with a *component inspector*
  - Can, in some environments, be connected to other components using visual metaphors
    - » Connecting input and output "ports" with "lines"
- Component is also accessible through programming (API):
  - Parameters can be manipulated by program code (getter, setter)
- There is a marketplace for components
  - Custom components
  - Building blocks for software

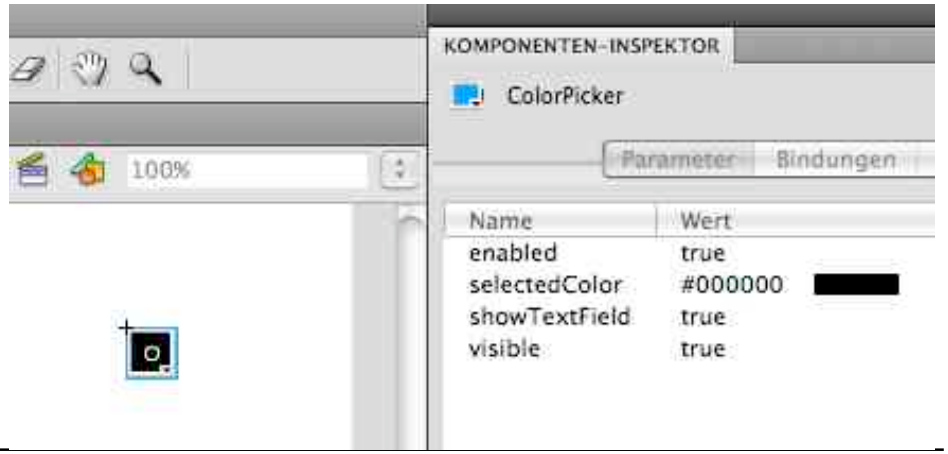
# Flash Components

- *Flash component:* A reusable unit of Flash design and ActionScript programming with clearly specified parameters and methods.
- A Flash component encapsulates a ready-made solution that can be incorporated into third-party Flash applications.
- Components delivered with Flash (CS4, examples):
  - User Interface components:
    - » Button, CheckBox, ColorPicker, ComboBox, DataGrid, Label, ProgressBar, ScrollPane, Slider, TextArea, TextInput, ...
  - Video components:
    - » FLVPlayback, ForwardButton, FullScreenButton, …
- File format for Flash components: *.swc*
  - Components are pre-compiled
  - Components cannot be edited by developer

# Example Flash Component: ColorPicker

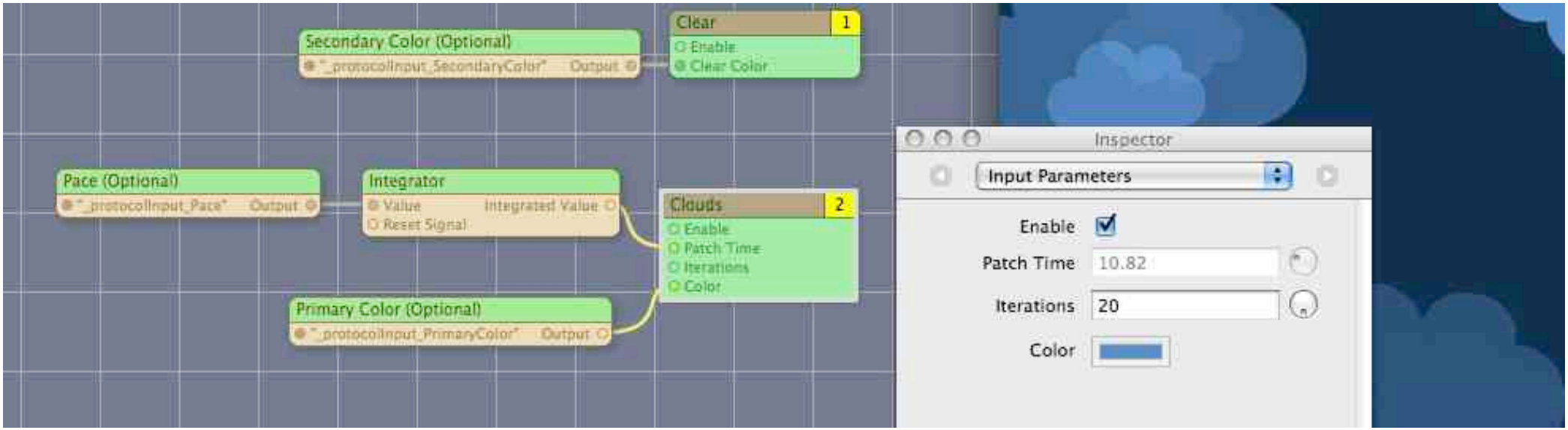
- Layout and basic behaviour are pre-defined
- Component inspector allows customization, e.g.
  - Definition of pre-selected color
  - Display of parts of dialog
- API allows dynamic ActionScript-based adaptation
  - E.g. enabling/disabling
- Components may generate events



# Advanced Component Composition

- Some authoring tools and development environments make it possible to graphically connect components
- Examples:
  - IBM Visual Age (Eclipse predeccessor)
  - Apple Quartz Composer (for graphics rendering)




Apple Quartz Composer

# 7 Programming with Video



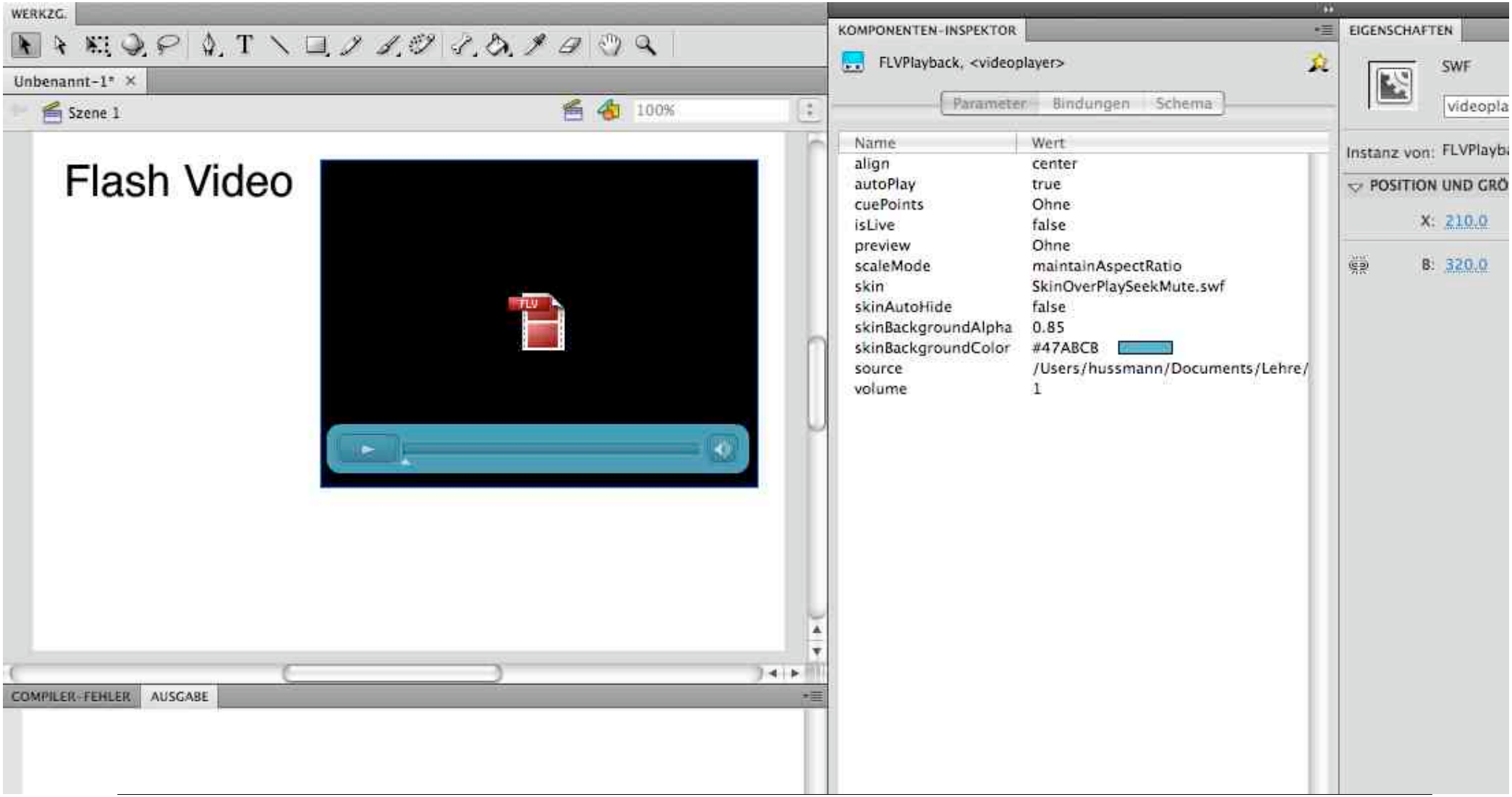
Literature:

Adobe Flash Documentation

# Playing Video in Flash

- Embedding video information into animation
  - Leads to very large files (SWF files in the case of Flash)
- External video clips:
  - Editable separately with specialized software
  - Progressive download: play during loading
  - Video played at its own frame rate, not at the rate of the animation
- Support for external video in Flash:
  - FLV (Flash Video) format (partially&recently also other formats)
  - Converters from most well-known video formats to FLV exist
  - Special *Media Components* for easy integration of video
    - » FLVPlayback
    - » Bars: BufferingBar, VolumeBar, SeekBar
    - » Buttons: PlayButton, PauseButton, MuteButton, …
    - » Captioning support

# Flash CS4 Video Player Component



# Themes/Skins

- Generally:
  - *Theme* or *skin* defines graphical appearance of interface elements
    - » Colours, number of elements, shapes, fonts, …
- For video playback component:
  - Skin defines playback control elements mainly
  - Which buttons exist, in which order, where located
- Predefined skins vs. Custom skins
- Examples (Flash):


# Example: Setting a Skin by Program Code

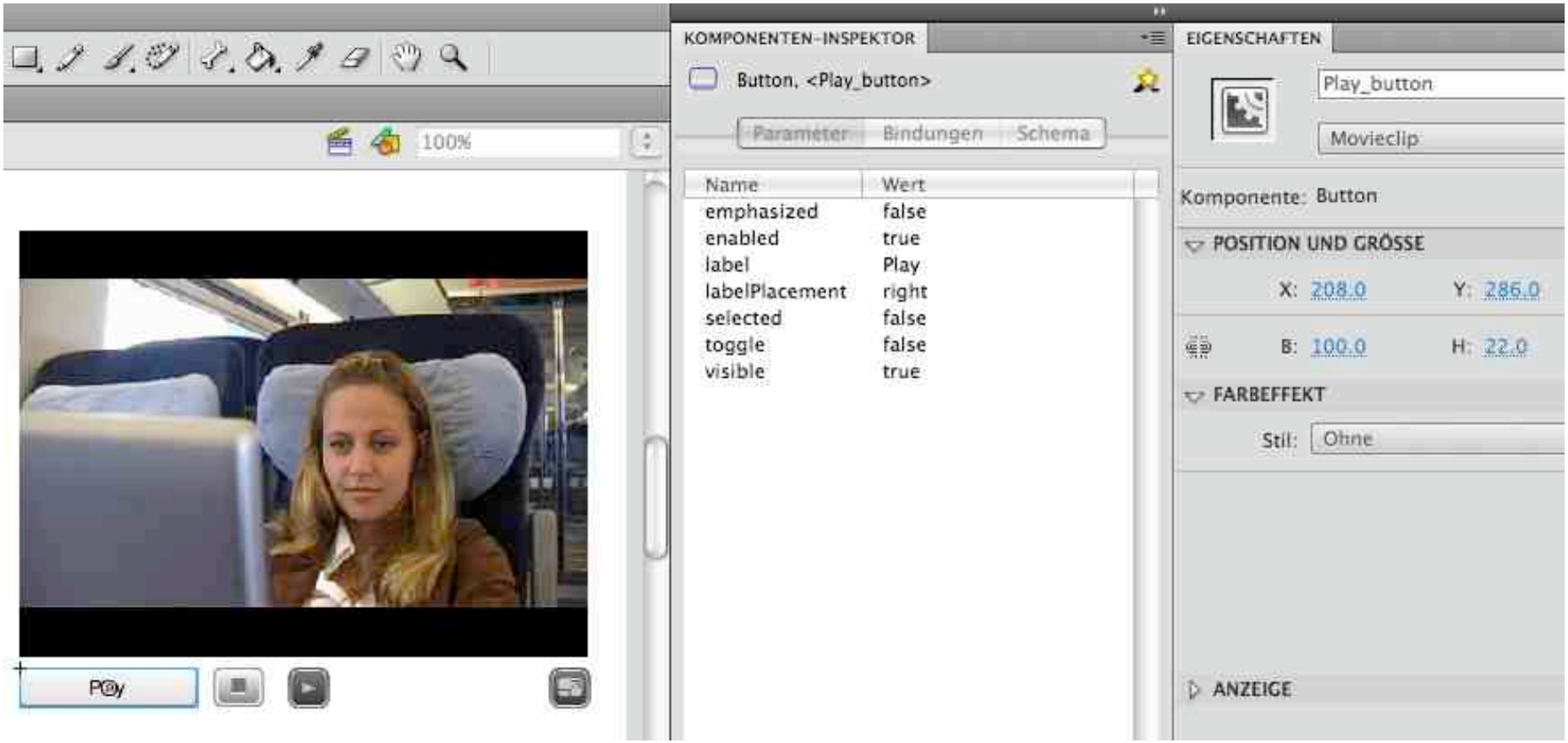
- ActionScript 3.0:

```
import fl.video.*;

var flvPlayer:FLVPlayback = new FLVPlayback();

addChild(flvPlayer);
flvPlayer.skin = "./SkinOverAll.swf“
flvPlayer.source = "../videos/OfficeCalling.f4v";
```

# Variants of Control Buttons

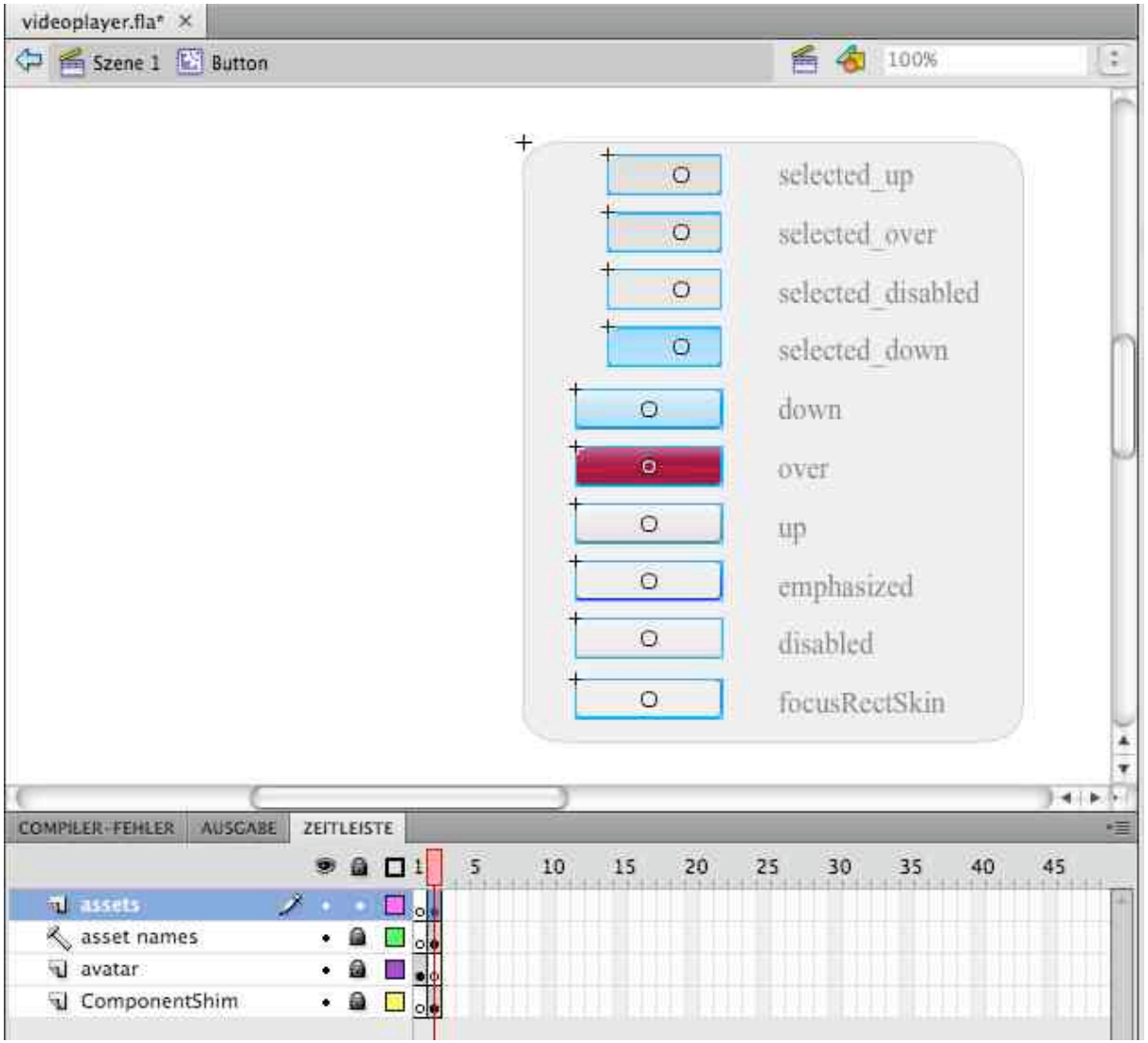
- Control buttons integrated into playback component
  - Possible adjustable by parameters or skins
- Special control button components
  - Flash: Video components “PauseButton”, “PlayButton” etc.
- Use of standard UI components
  - Customized for specific purpose
- Use of pre-designed UI components
  - Bought from external source
  - Loaded from external library

# Customizing a Button for Video Control (1)



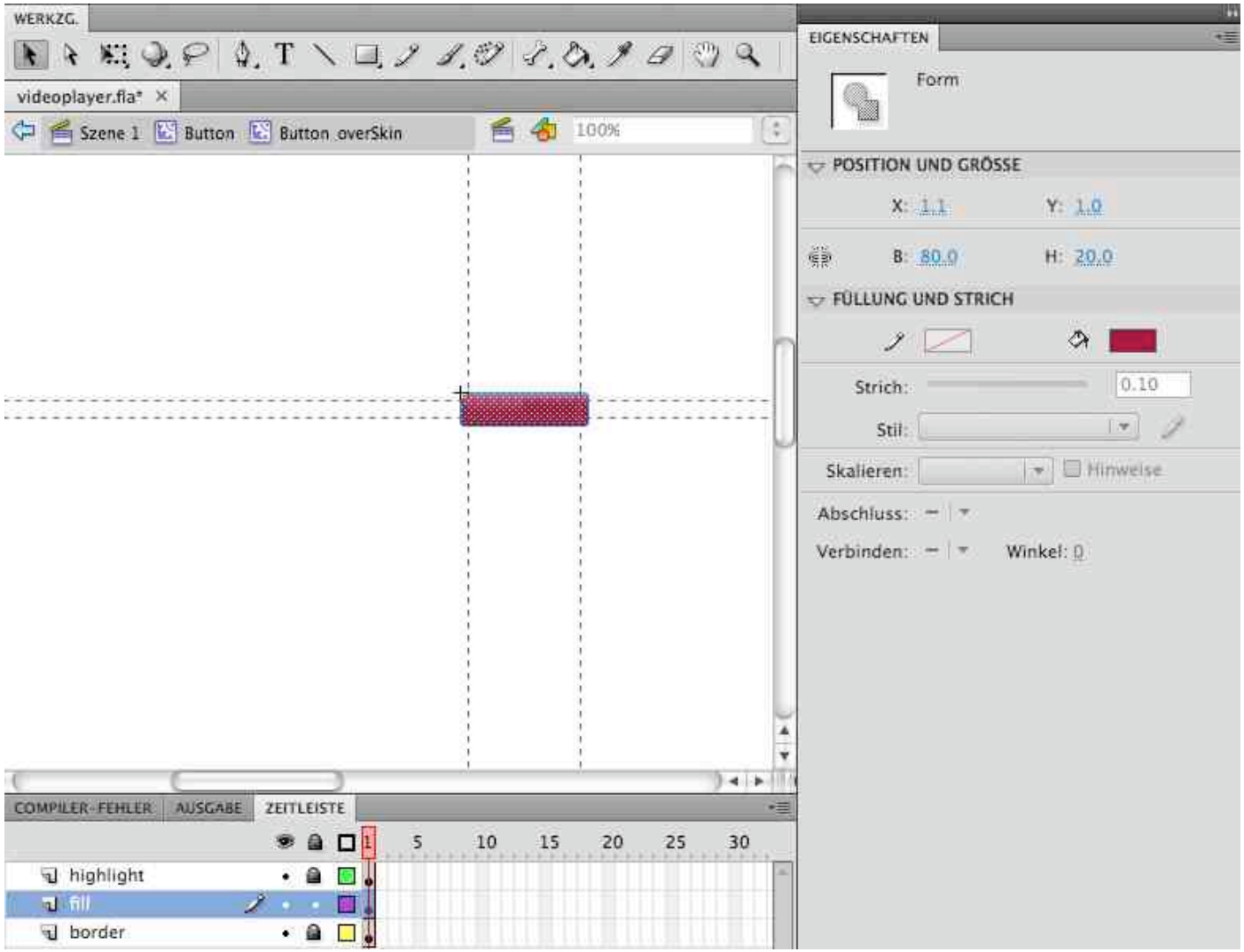

- Standard UI component: Button
  - Drag&drop, label parameter set

# Customizing a Button for Video Control (2a)



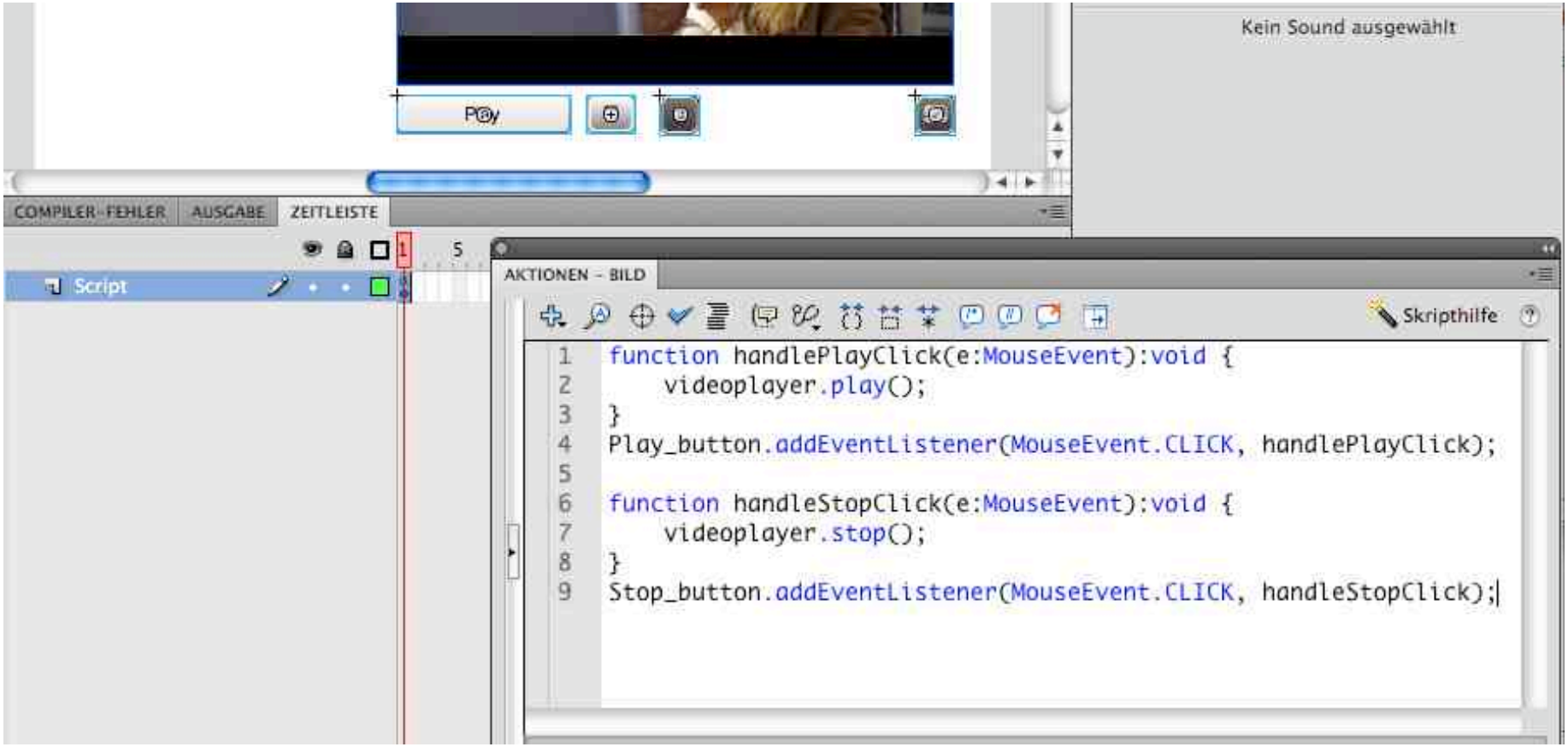

- Timeline of Button component
  - Contains nested graphics for state visualization
- An example for fully graphical component customization

# Customizing a Button for Video Control (2b)



# Customizing a Button for Video Control (3)

- Adding event handler as script code
  - References to component instances



# 7 Programming with Video

7.1 Components for Multimedia Programs

7.2 Video Player Components

7.3 Interactive Video 

7.4 Integrating Video into Web Pages

Literature:

Florian Plag, Roland Riempp: Interaktives Video im Internet mit Flash, Springer 2006 (und www.video-flash.de)

# Events Generated by Media Components

- Various events are reported by Media Components to the surrounding application for flexible reaction:
  - Adjustments like change of volume
  - Media events like reaching end of media
  - User-defined events when reaching specific positions *(cue events)*

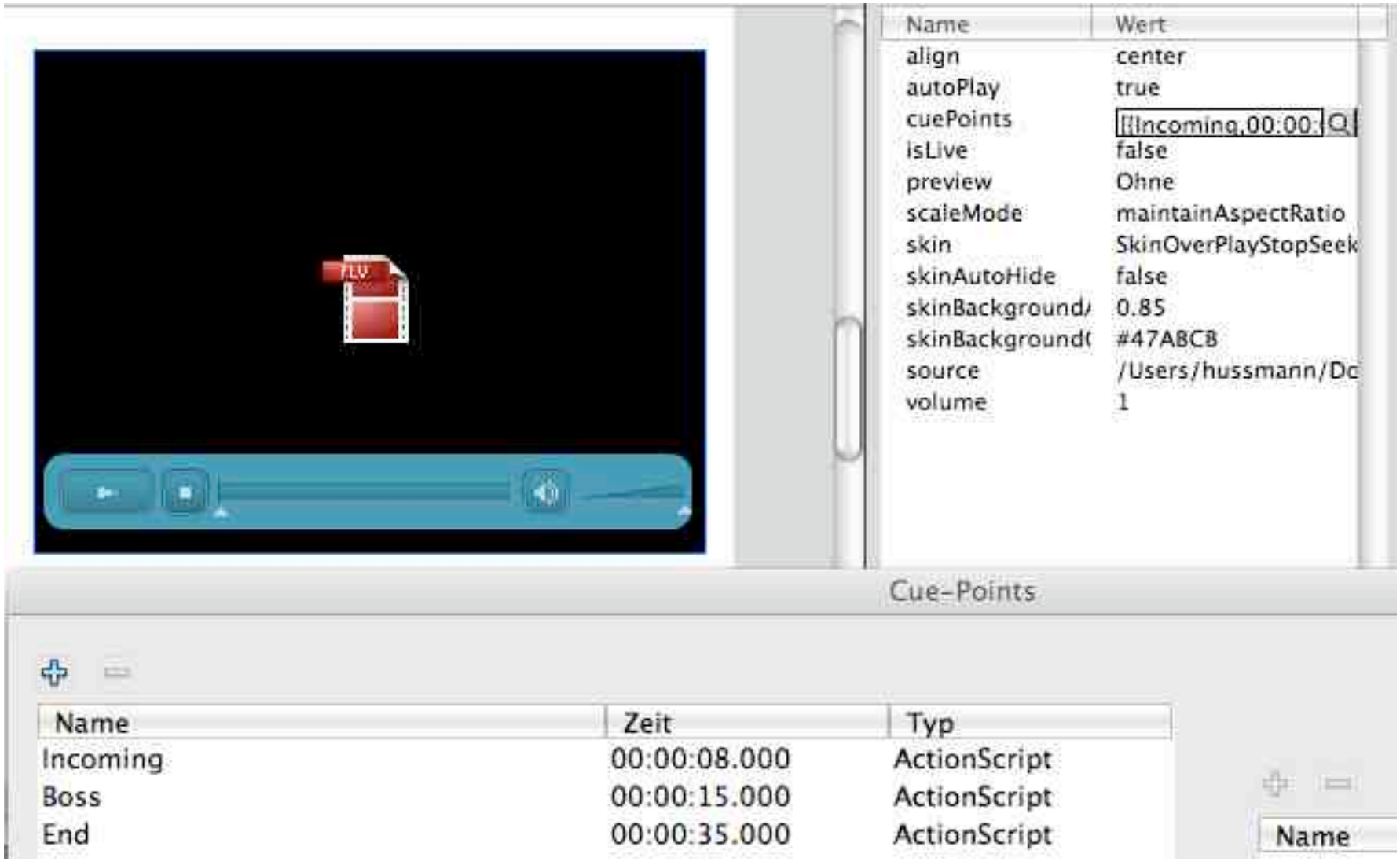
- Reaction to media events requires *EventListener* objects for media specific events, e.g. **`SoundEvent`**:

```
function handleSoundUpdate(e:SoundEvent):void {
    trace("volume/pan changed");
videoplayer.addEventListener
    (SoundEvent.SOUND_UPDATE, handleSoundUpdate);
```

# Cue Points

- A *cue point* marks a specific point in time during media playback.
  - Specification by *time stamp* relative to media start time
  - Definition in authoring system: Interactively or by numbers
  - May be defined by script code
- Internal cuepoint: Embedded into movie file
  - Flash: navigation (for direct positioning) and event cuepoints (for interaction with web browser)
- External cuepoint: Defined outside movie file
  - Flash: ActionScript cuepoints
  - When reaching a cue point, an (AS) event is fired

# Cue Points in Authoring System

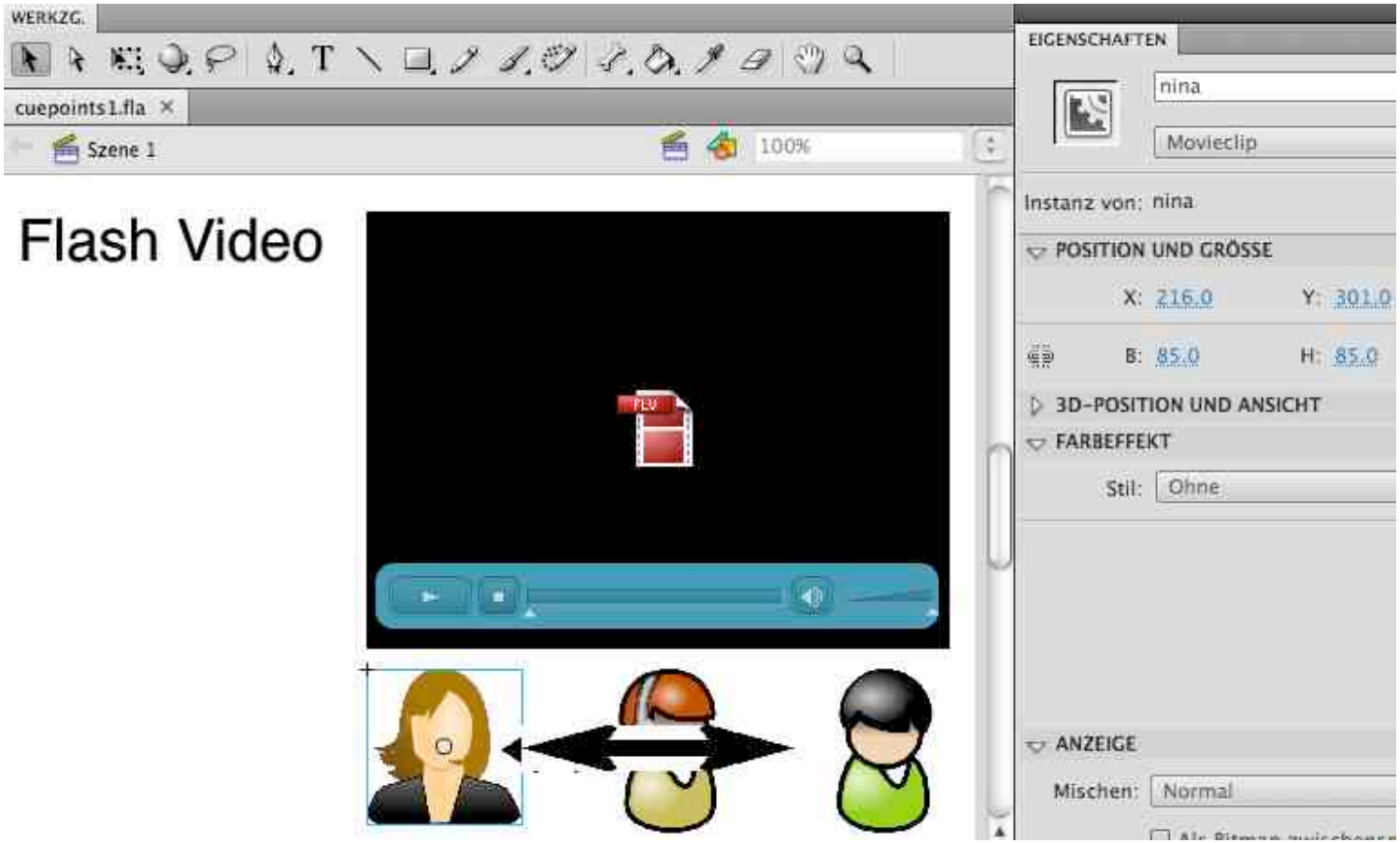


# Cue Points Defined by Script

```
var cuePt1:Object = new Object();
cuePt1.time = 0.08;
cuePt1.name = “Incoming“;
cuePt1.type = "actionscript";
videoplayer.addASCuePoint(cuePt1);

videoplayer.addASCuePoint(0.15, “Boss");
```

# Synchronizing Video and Animation (1)

- Cue point events can trigger animation actions
- Simple example: Visibility of symbolic annotations



# Synchronizing Video and Animation (2)

```
sekr.visible = false;
boss.visible = false;
pfeil1.visible = false;
pfeil2.visible = false;

videoplayer.addEventListener
   (MetadataEvent.CUE_POINT, cpT_listener);
function cpT_listener(e:MetadataEvent):void {
   if (e.info.name == "Incoming") {
       sekr.visible = true;
       pfeil1.visible = true;
   }
   if (e.info.name == "Boss") {
       sekr.visible = false;
       boss.visible = true;
       pfeil2.visible = true;
       pfeil1.visible = false;
   }
   if (e.info.name == "End") {
       boss.visible = false;
       pfeil2.visible = false;
   }
}
```

# How to Realize Real Interaction in Video?

- Real interaction means:
  - Pointing to regions in video window identifies objects
  - Clicking on a region or symbol modifies video scene
- Scene needs to be *decomposed:*
  - Parts/objects of video playback can be made (in)visible by script code
  - Objects can be moved around in video
- Easy solution:
  - *Overlaying* of videos
- Two main techniques:
  - *Masking* cuts out specific parts from a video
    - » Prerequisite: Objects are easy to identify and do not move much
  - *Alpha channel* videos overlayed on other videos
    - » Prerequisite: External production of video with alpha channel
    - » Using video effect software (e.g. AfterEffects)

# Application Examples (1)

**#01: Gipsy Voices**

Ein Hauptmenü in Form einer leeren Bühne lädt den Nutzer dazu ein, spielerisch zu erkunden, welche Musiker in der Band "Gipsy Voices" spielen. Klickt der Anwender eine Person an, öffnet sich ein Fenster, das mehrere Videos zur jeweiligen Person bietet.

www.video-flash.de

# Application Examples (2)

**#02: Hutshop**
Dieser Entwurf soll einen Eindruck vermitteln, wie eine Produktpräsentation mit Videos im Internet ausssehen könnte. Es werden freigestellte Videos verwendet.

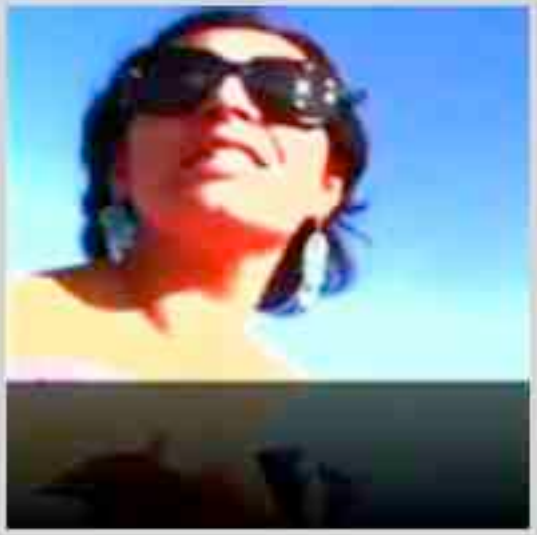
www.video-flash.de

# Application Examples (3)


**#08: Hot-Spots**
Beim Encoding des verwendeten Videos wurden Cue-Points eingebettet, die beim Abspielen der Anwendung die Bezeichnung und die Position eines Hot-Spots bestimmen. Die weiterführenden SWF-Dateien werden ebenfalls in Abhängigkeit des CuePoint-Namens nachgeladen.

www.video-flash.de

# Application Examples (4)



**#15: Reflektionen**
Ein schöner Effekt ist die Erzeugung von Reflektionen, wodurch die Anmutung einer Rich-Media-Anwendung erhöht wird. In diesem Beispiel wird ständig das aktuelle Videobild mithilfe der BitmapData-Klasse dupliziert, dann gespiegelt und unterhalb des Videos wieder eingefügt. Eine halbtransparente Maske sorgt für ein weiches Ausblenden der Reflektion.

www.video-flash.de

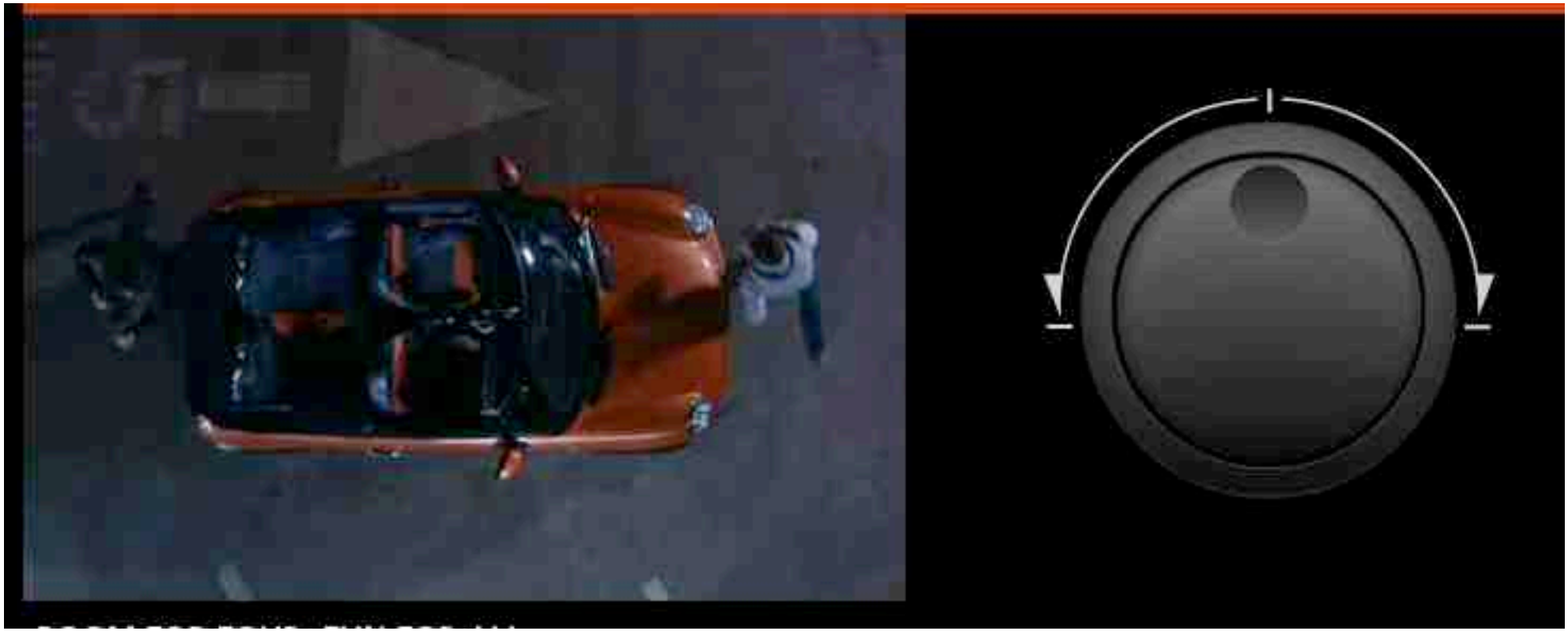
# Application Examples (5)

**#18: Fernsehgerät**
Über eine Maskierung wird das Video in die Abbildung des TV-Geräts eingepasst. Das schwarz-weiße Rauschen des TVs kann mit der „Noise"-Funktion der Bitmap-Klasse erstellt werden, die ein Pixelbild mit zufälligen Störungen erzeugt.

www.video-flash.de

# Application Examples (6)

http://www.mini.com/com/en/mini_cabrio_film_clips/index.jsp



# 7 Programming with Video

7.1 Components for Multimedia Programs

7.2 Video Player Components

7.3 Interactive Video

7.4 Integrating Video into Web Pages 

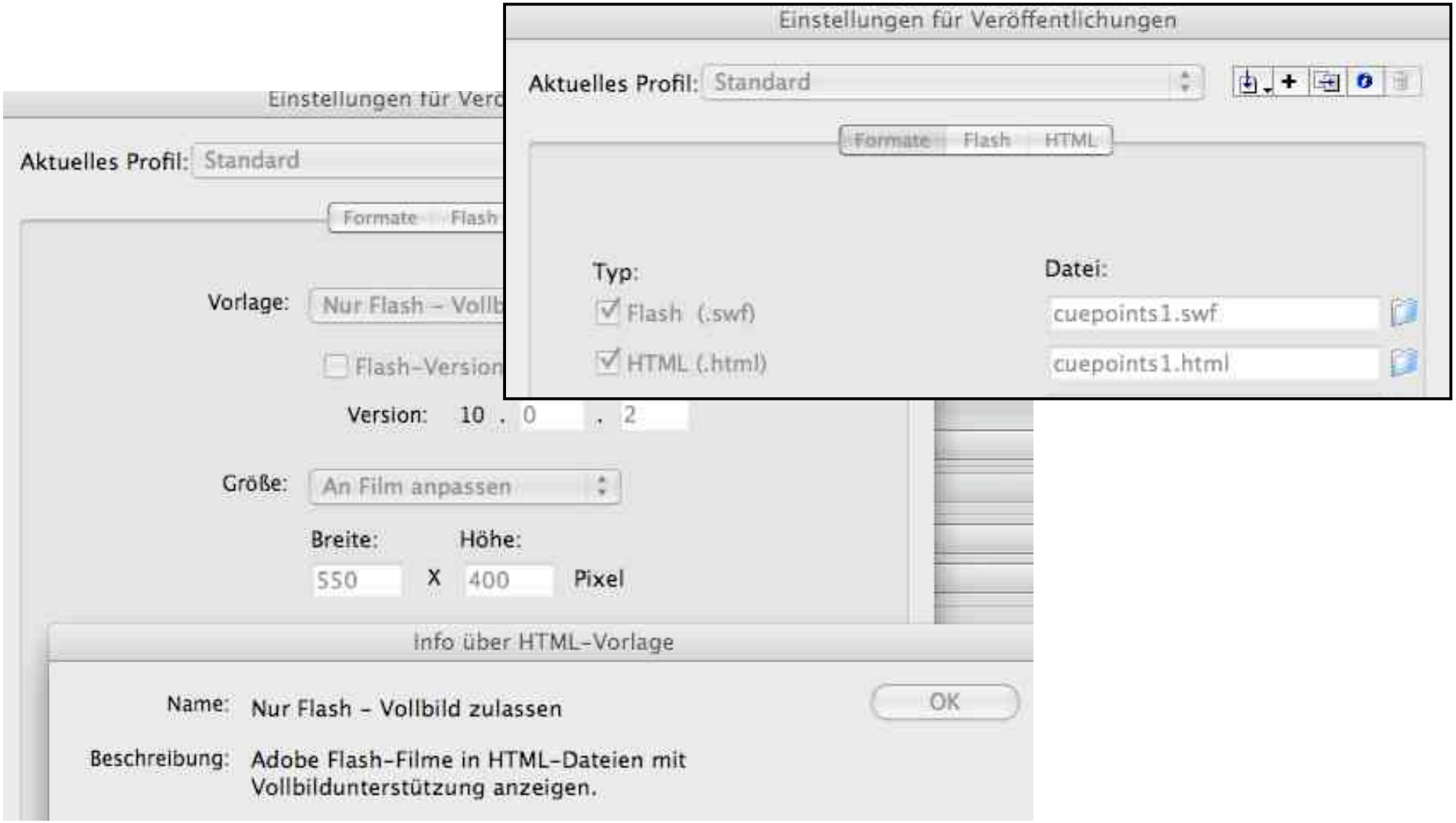
Literature:

Adobe Documentation for Flash

# Interactive Multimedia into Web Pages

- Interactive multmedia requires *player* software
  - Available as plugins for popular Web browsers
- Web page (HTML) loaded into browser
  - Needs to adapt actively: JavaScript code
  - Tasks:
    - » Identification of browser and operating system
    - » Identification of installed player software and version
    - » Creation of adequate HTML code
      - embed/object tags
    - » Passing parameters to player
      - E.g. on usage of full-screen mode

# HTML Support in Authoring System



# Excerpt of Generated HTML Code (1)

```
<script language="JavaScript" type="text/javascript">
<!--
//v1.7
// Flash Player Version Detection
// Detect Client Browser type
// Copyright 2005-2008 Adobe Systems Incorporated.  All
  rights reserved.
var isIE  = (navigator.appVersion.indexOf("MSIE") != -
  1) ? true : false;
var isWin =
  (navigator.appVersion.toLowerCase().indexOf("win") !=
  -1) ? true : false;
var isOpera = (navigator.userAgent.indexOf("Opera") !=
  -1) ? true : false;
function ControlVersion()
{
  var version;
  ...
}
```

# Excerpt of Generated HTML Code (2) (Flash)

```
function GetSwfVer(){
  var flashVer = -1;

  if (navigator.plugins != null
    && navigator.plugins.length > 0) {
      if (navigator.plugins["Shockwave Flash 2.0"] ||
        navigator.plugins["Shockwave Flash"]) {
            var swVer2 =
              navigator.plugins
                ["Shockwave Flash 2.0"] ? " 2.0" : "";
            var flashDescription =
                navigator.plugins["Shockwave Flash" +
                  swVer2].description;
            var descArray = flashDescription.split(" ");
            var tempArrayMajor = descArray[2].split(".");

            var versionMajor = tempArrayMajor[0];
            var versionMinor = tempArrayMajor[1];
...
            var flashVer =
              versionMajor + "." + versionMinor + "."
                + versionRevision;
      }
```

# Excerpt of Generated HTML Code (3)

```
function AC_Generateobj(objAttrs, params, embedAttrs)
{
  var str = '';
  if (isIE && isWin && !isOpera)
  {
    str += '<object ';
    for (var i in objAttrs)
    {
      str += i + '="' + objAttrs[i] + '" ';
    }
    str += '>';
    for (var i in params)
    {
      str += '<param name="' + i + '" value="' + params[i] + '" /> ';
    }
    str += '</object>';
  }
  else
  {
    str += '<embed ';
    for (var i in embedAttrs)
    {
      str += i + '="' + embedAttrs[i] + '" ';
    }
    str += '> </embed>';
  }
  document.write(str);
}
```