PEARSON Education

ADDISON-WESLEY

Selma-Caroline Kannengiesser
Matthias Kannengiesser

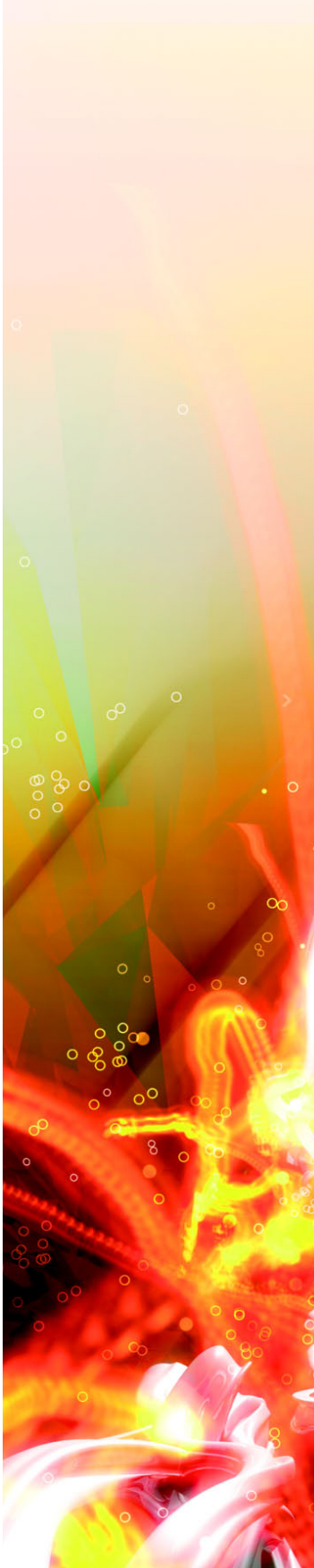
Flash 8

Powerworkshops

Kapitel 3

Dynamische Inhalte in Flash 8

In diesem Kapitel wenden wir uns den dynamischen Inhalten zu und stellen Ihnen einige interessante Anwendungen vor. Wir sind uns sicher, die diversen Anregungen werden Ihre Fantasie beflügeln.



3.1 Dynamisches Profil – Textinhalte und CSS-Stile

Im folgenden Workshop wollen wir Ihnen eine Möglichkeit vorstellen, wie einfach es sein kann, ein dynamisches Profil anzulegen, das mit Hilfe einer einfachen Textdatei und einer CSS-Datei individuell angepasst werden kann.

Schritt 1:

Wir benötigen als Erstes einen Flash-Film, diesen sollten Sie unter dem Namen *CSS_Profil.fla* speichern. Aus dem Komponenten-Bedienfeld, das Sie mit Hilfe der Option Fenster/Komponenten erreichen, ziehen Sie eine Instanz der TextArea-Komponente auf die Bühne.

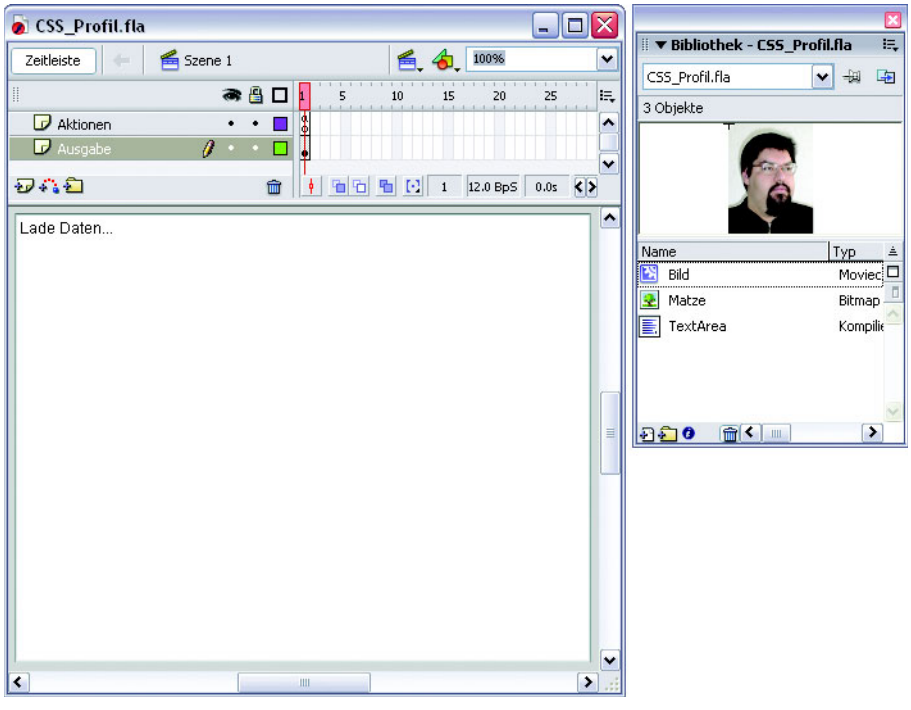


Abbildung 3.1: TextArea-Komponente samt benötigter Symbole

Schritt 2:

Wählen Sie die TextArea-Komponente und weisen Sie ihr den Instanznamen `display_txt` zu. Wählen Sie anschließend aus der Bibliothek das MovieClip-Symbol *Bild* aus und weisen diesem den Verknüpfungsbezeichner `bild` zu.

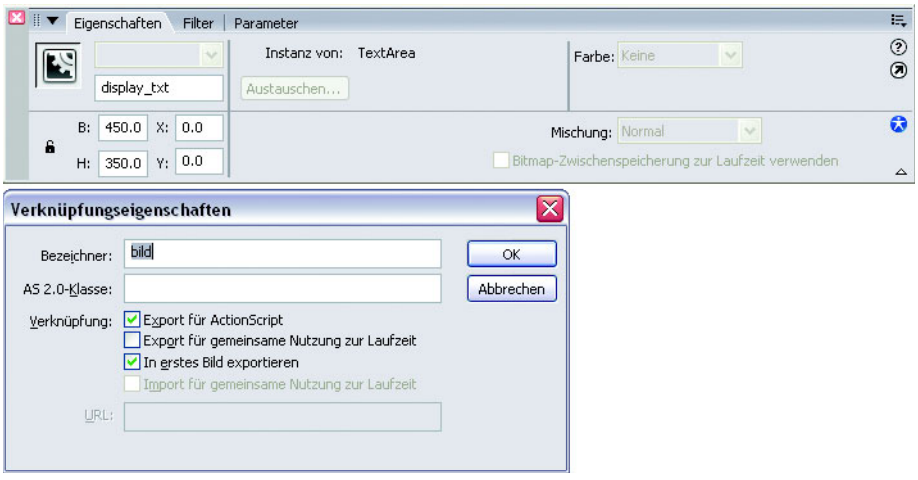


Abbildung 3.2: Instanzname der TextArea-Komponente und der Verknüpfungsbezeichner des Bild-Movieclip-Symbols

▶ Schritt 3:

Als Nächstes sollten Sie sich um den Inhalt des Profils kümmern. Hierfür benötigen Sie eine Textdatei, die ein *HTML*-Grundgerüst enthält und unter dem Namen *inhalt.html* gespeichert werden sollte.

```
<body>
  <br />
  <textformat leftmargin='30' rightmargin='30'>
    <headline>Profil: Matthias Kannengiesser
    ➥ </headline>
    <img align='left' src='bild' width='100'
    ➥ height='130' hspace='30' vspace='20' />
    <br />
    <details>
      Name: <span class="infoText">
      ➥ Matthias Kannengiesser</span><br />
      Alter: <span class="infoText">31</span><br />
      Beruf: <span class="infoText">
      ➥ Informatiker</span><br />
      Hobbies: <span class="infoText">Girls &amp;
      ➥ Nature</span><br />
      Wohnort: <span class="infoText">
      ➥ Berlin/Germany</span>
    </details>
    <br /><br /><br />
    <inhalt>
      ➥ Die Buch-CD stellt eine zus&auml;tzliche
      ➥ Fundgrube dar und erspart Ihnen die
      ➥ zeitraubende Suche nach n&uuml;tzlichen
      ➥ Tutorials, Beispielen und Tools aus dem
      ➥ unendlich grossen Ozean des WorldWideWeb.
      ➥ Wir empfehlen Ihnen sowohl das Buch, als
      ➥ auch die CD einzusetzen, um erfolgreich
      ➥ mit Flash 8 Basic und Flash 8 Professional
      ➥ durchzustarten.<br /><br />
      <a href="http://www.flashstar.de" target=
      ➥ "_blank">Flashstar Website Project</a>
      ➥ oder <a href="mailto:matzek@flashstar.de"
      ➥ target="_blank">E-mail</a>!
    </inhalt>
  </textformat>
</body>
```

▶ Schritt 4:

Eine Formatierung sollen die jeweiligen Abschnitte der Textdatei natürlich auch erhalten. Diese Aufgabe übernehmen die *CSS*-Stilzuweisungen, die in der Datei *mein.css* platziert werden.

```
headline {
  font-family: Arial,Helvetica,sans-serif;
  font-size: 16px;
  font-weight: bold;
  display: block;
  color: #FF9900;
  text-decoration:underline;
}
details {
  font-family: Arial,Helvetica,sans-serif;
  font-size: 12px;
  font-weight: bold;
  color: #FF9900;
  display: block;
}
inhalt {
  font-family: Arial,Helvetica,sans-serif;
  font-size: 12px;
  font-weight: bold;
  color: #000000;
  display: block;
}
a {
  font-family: Arial,Helvetica,sans-serif;
  color: #0066FF;
  font-size: 12px;
  display: inline;
  text-decoration:underline;
}
.infoText {
 color: #0066FF;
}
```

▶ Schritt 5:

Im ersten Schlüsselbild der Zeitleiste platzieren Sie in der *Aktionen*-Ebene folgende ActionScript-Codezeilen.

```
// Initialisierung - TextArea-Komponente
display_txt.html = true;
display_txt.wordWrap = true;
display_txt.multiline = true;
display_txt.label.condenseWhite = true;

// Hintergrund der TextArea-Komponente auf
// unsichtbar
_global.styles.TextArea.setStyle
➥ ("backgroundColor", "false");

// CSS-Datei (laden) und mit der TextArea verknüpfen
var meinStyle:TextField.StyleSheet =
➥ new TextField.StyleSheet ();
meinStyle.load ("mein.css");
display_txt.styleSheet = meinStyle;

// XML-Datei (laden) und den Inhalt der TextArea
// zuweisen
var inhalt_xml:XML = new XML ();
inhalt_xml.ignoreWhite = true;
inhalt_xml.load ("inhalt.html");
inhalt_xml.onLoad = function (success){
  if (success)
  {
    display_txt.text = inhalt_xml;
  }
};
```

▶ Schritt 6:

Sie können den Film testen und mit Hilfe der *CSS*-Datei die Stilzuweisungen für die einzelnen Abschnitte nach Ihren Wünschen anpassen.

Abbildung 3.3: Interpretation des Flash Players (HTML und CSS-Stil)

Anregung:

Sollte Ihnen die Lösung noch nicht dynamisch genug sein, könnten Sie ja die beiden Symbole *Bild* und *Matze* aus der Bibliothek entfernen und das Bild ebenfalls dynamisch einlesen. Hierfür muss lediglich eine kleine Anpassung an der *HTML*-Datei (Textdatei) vorgenommen werden:

```
<img align='left' src='bilder/matze.png'
width='100' height='130' hspace='30' vspace='20' />
```

Wie Sie sehen haben wir in diesem Fall auf eine externe *PNG*-Datei verwiesen. Mit dem `<img>`-Tag können Sie *JPEG*-, *GIF*-, *PNG*- und *SWF*-Dateien einlesen und in Textfeldern bzw. TextArea-Komponenten-Instanzen einbetten. Textinhalte fließen hierbei automatisch um diese Objekte oder Komponenten herum. Um dieses Tag verwenden zu können, müssen Sie das dynamische Feld oder Eingabetextfeld als mehrzeilig definieren und Zeilenumbrüche aktivieren.

Das `<img>`-Tag unterstützt die folgenden Attribute:

- src Definiert die URL zu einer Bild- oder SWF-Datei bzw. den Verknüpfungsbezeichner eines Movieclip-Symbols in der Bibliothek. Dieses Attribut ist erforderlich. Alle weiteren sind optional. Externe Dateien werden erst dann angezeigt, wenn sie komplett heruntergeladen worden sind.
- id Definiert den Namen der Movieclip-Instanz (wird vom Flash Player erstellt), welche die eingebettete Bild- oder SWF-Datei bzw. den Movieclip enthält. Dies ist nützlich, wenn Sie den eingebetteten Inhalt mit ActionScript steuern möchten.
- width Die Breite des einzufügenden Bilds, der SWF-Datei oder des Movieclips in Pixel.
- height Die Höhe des einzufügenden Bilds, der SWF-Datei oder des Movieclips in Pixel.
- align Legt die horizontale Ausrichtung des eingebetteten Bilds im Textfeld fest. Gültige Werte sind `left` und `right`. Der Standardwert lautet `left`.
- hspace Legt die Größe des horizontalen Bereichs um das Bild fest, in dem kein Text zu sehen ist. Der Standardwert ist 8.
- vspace Legt die Größe des vertikalen Bereichs um das Bild fest, in dem kein Text zu sehen ist. Der Standardwert ist 8.

3.1.1 Dynamische Bilder – GIF, JPEG und PNG

Im folgenden Workshop wollen wir Ihnen die Möglichkeit vorstellen, wie einfach es ist dynamische Bilder in eigene Flash-Produktionen zu integrieren. Es stehen Ihnen nun auch weit mehr *Bitmap*-Dateiformate zur Verfügung. Vorbei die Zeiten, wo lediglich die Integration von dynamischen *Standard-JPEG*-Bildern in Flash-Produktionen möglich war. Flash 8 ermöglicht es Ihnen *Standard-/Progressiv-JPEG*-, *GIF*- und *PNG*-Bilder zu integrieren, sprich die gängigen *Bitmap*-Dateiformate, die sonst auch auf *HTML*-basierten Webseiten zum Einsatz kommen.

▶ Schritt 1:

Wir benötigen als Erstes einen Flash-Film, diesen sollten Sie unter dem Namen *Externe_Bitmaps.fla* speichern. Die Bühne sollte *640* Pixel breit und *484* Pixel hoch sein und die Bildrate kann beliebig gewählt werden. Die Bühnendimensionen werden natürlich durch die zu ladenden Bilder vorgegeben, daher sollten Sie sich nicht wundern, dass die Höhe etwas aus dem Rahmen fällt.

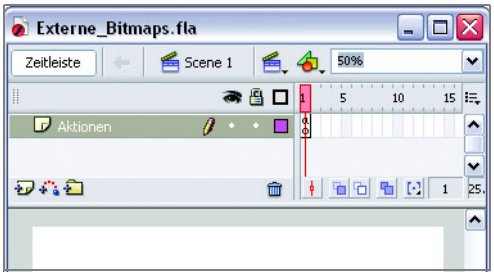


Abbildung 3.4: Aufbau des Flash-Films – lediglich eine Aktionen-Ebene wird benötigt

Schritt 2:

Im ersten Schlüsselbild der Zeitleiste platzieren Sie in der *Aktionen*-Ebene folgende ActionScript-Codezeilen.

```
Stage.scaleMode = "noScale";

// Liste der externen Bilder
var bilder:Array = ["Bilder/Sonnenblume.jpg",
➥ "Bilder/FlashLogo.png",
➥ "Bilder/logo.gif"];

for (var i = 0; i < bilder.length; i++)
{
  // Jedem Bild eine eigene Movieclip-Instanz
  // zuweisen
  var clip:MovieClip = this.createEmptyMovieClip
  ➥ ("halter" + i, (i + 1));
  clip.createEmptyMovieClip ("container", 1);
  clip.container.loadMovie (bilder[i]);
  // Drag & Drop-Verhalten
  clip.onPress = function ()
  {
    this.startDrag ();
  };
  clip.onRelease = clip.onReleaseOutside =
  ➥ function ()
  {
    this.stopDrag ();
  };
}
```

Schritt 3:

Sie können den Film testen und wie Sie sehen wurde jeder Movieclip-Instanz ein Drag & Drop-Verhalten zugewiesen und, was besonders auffällig ist, der Alphakanal der *PNG*-Datei wird vom Flash Player korrekt wiedergegeben, so dass sich daraus beispielsweise neue Möglichkeiten für die Maskierung von Inhalten ergeben.

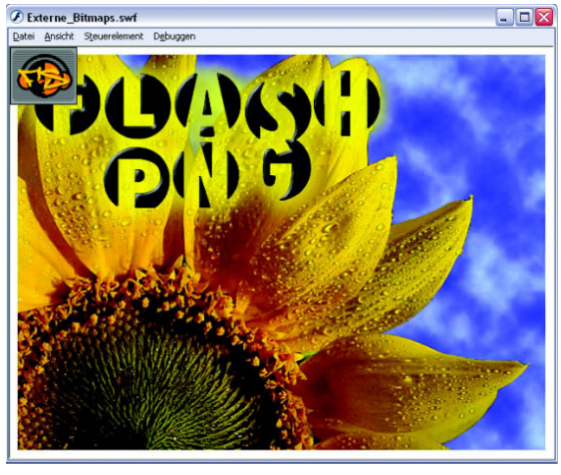


Abbildung 3.5: Externe Bitmaps in Echtzeit eingelesen

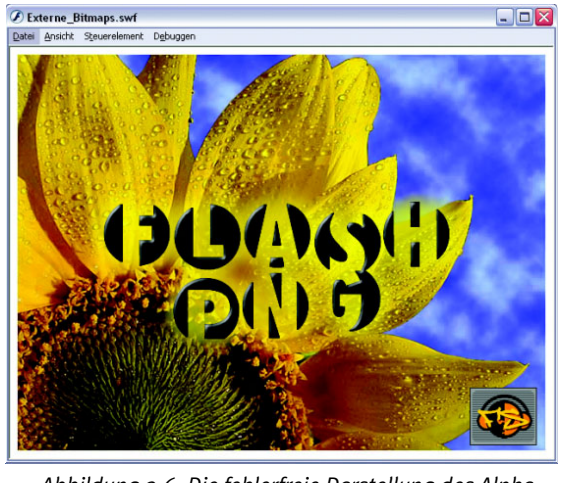


Abbildung 3.6: Die fehlerfreie Darstellung des Alphakanals von PNG-Dateien stellt kein Problem dar.

3.1.2 Dynamische Bildgalerie auf Basis von XML

Im folgenden Workshop geht es um die Umsetzung einer dynamischen Bildgalerie, deren Inhalte sich mit Hilfe einer separaten *XML*-Datei festlegen lassen.

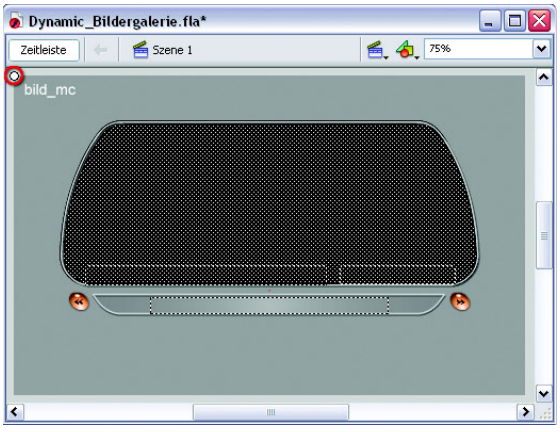


Abbildung 3.8: Die Movieclip-Instanz bild_mc befindet sich in der maskierten BildClip-Ebene.

▶ **Schritt 1:**

Wir benötigen als Erstes einen Flash-Film, diesen sollten Sie unter dem Namen *Dynamic_Bildergalerie.fla* speichern. Die Bühne sollte *640* Pixel breit und *400* Pixel hoch sein.

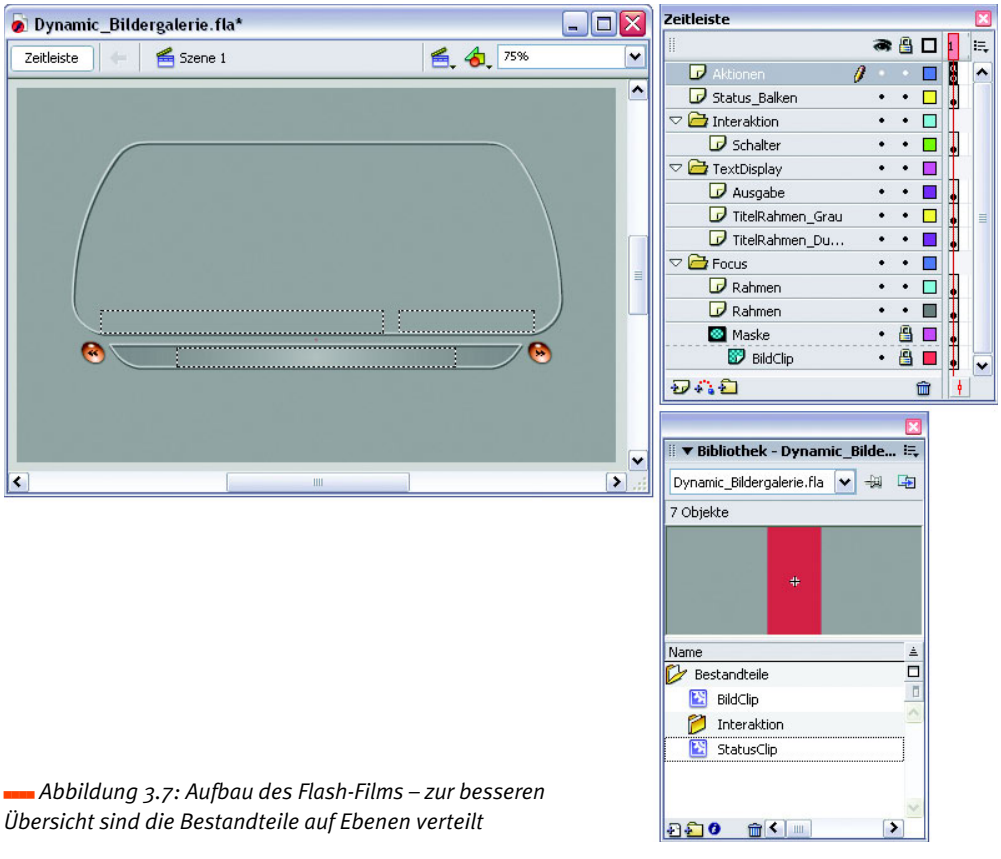


Abbildung 3.7: Aufbau des Flash-Films – zur besseren Übersicht sind die Bestandteile auf Ebenen verteilt

▶ Schritt 2:

Sie benötigen drei Textfelder, denen Sie die Instanznamen `loading_txt`, `counter_txt` und `titel_txt` zuweisen. Darüber hinaus werden zwei Movieclip-Instanzen mit den Instanznamen `bild_mc` und `status_mc` benötigt. Die Schlusslichter stellen zwei Schaltflächen-Instanzen `weiter_btn` und `zurück_btn` dar.

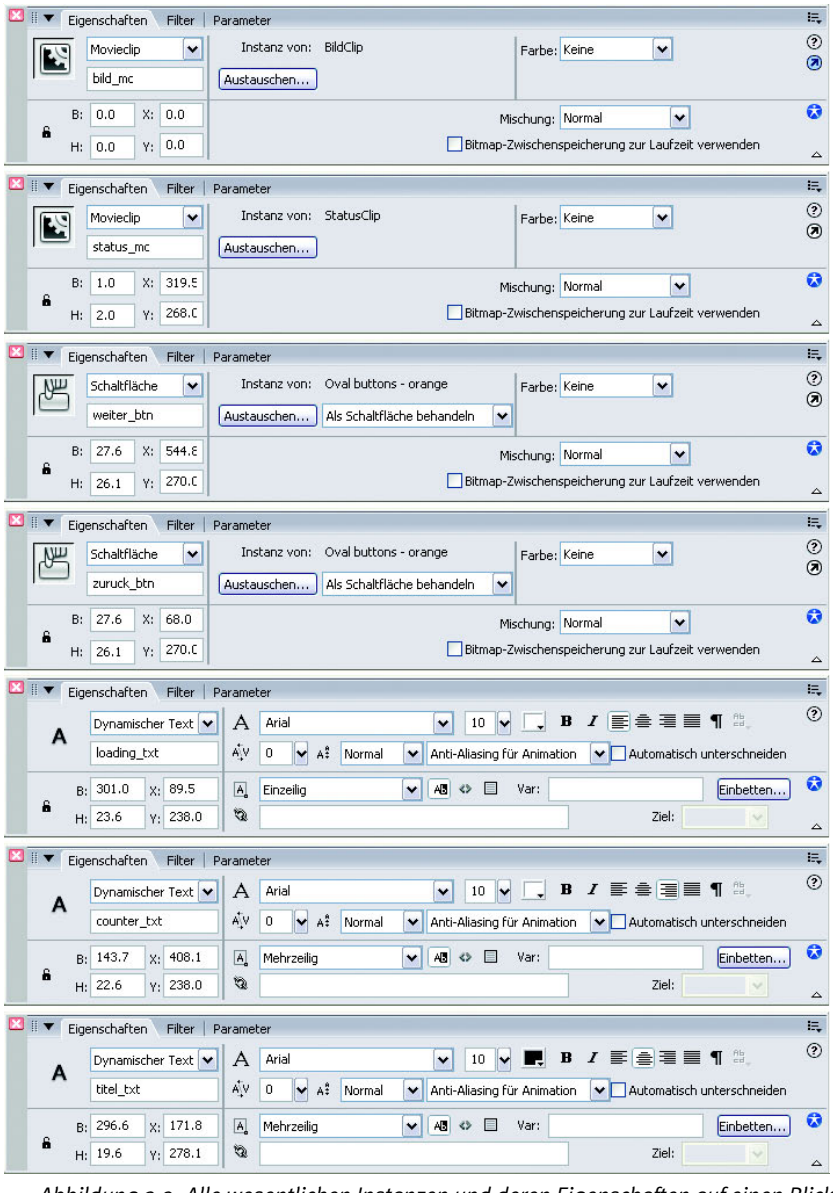


Abbildung 3.9: Alle wesentlichen Instanzen und deren Eigenschaften auf einen Blick

Schritt 3:

Als Nächstes sollten Sie sich um den Inhalt kümmern. Hierfür benötigen Sie eine *XML*-Datei, die die Bilddaten enthält und unter dem Namen *bilddaten.xml* gespeichert werden sollte.

```
<?xml version"1.0"?>
<Bilder>
<Bild name="Man on Mars - Lunatic Lander"
➥ pic="bilder/bild1.jpg"></Bild>
<Bild name="Pluto City" pic="bilder/bild2.jpg">
➥ </Bild>
<Bild name="Desert of Venus"
➥ pic="bilder/bild3.jpg"></Bild>
<Bild name="Working Mines on Planet Mars"
➥ pic="bilder/bild4.jpg"></Bild>
<Bild name="Movement of the Terraformer"
➥ pic="bilder/bild5.jpg"></Bild>
</Bilder>
```

Schritt 4:

Im ersten Schlüsselbild der Zeitleiste platzieren Sie in der *Aktionen*-Ebene folgende ActionScript-Codezeilen.

```
// XML-Objekt erzeugen (Bezeichner: mein_xml)
mein_xml = new XML ();
// Störende Leerzeichen ignorieren
mein_xml.ignoreWhite = true;
// XML-Datei einlesen
mein_xml.load ("bilddaten.xml");
// Lesevorgang
mein_xml.onLoad = function (status){
  if (status && this.loaded)
  {
    anzahl = 0;
    anzahl = mein_xml.firstChild.childNodes.length;
    geladen = true;
    aktBild = 0;
    ladeBild (aktBild);
  }
};

// Funktion, welche die Daten der XML-Datei
// verarbeitet
function ladeBild (paktBild){
  status_mc._visible = 1;
  bild = mein_xml.firstChild.childNodes
  ➥ [paktBild].attributes.pic;
  bildname = mein_xml.firstChild.childNodes
  ➥ [paktBild].attributes.name;
  loadMovie (bild, "bild_mc");
  titel_txt.text = bildname;
}

// Weiter-Schalter
weiter_btn.onRelease = function (){
  if (geladen && aktBild < anzahl - 1)
  {
    aktBild++;
    ladeBild (aktBild);
  }
};

// Zurück-Schalter
zuruck_btn.onRelease = function (){
  if (geladen && aktBild > 0)
  {
    aktBild--;
    ladeBild (aktBild);
  }
};
```

▶ Schritt 5:

Auf der Movieclip-Instanz `bild_mc` platzieren Sie folgende ActionScript-Codezeilen.

```
onClipEvent (load) {
  var i = 0;
}
onClipEvent (enterFrame) {
  if (this._url != this._parent._url
  ➥ && !this.loaded)
  {
    var kilobytes = Math.ceil (this.getBytesTotal ()
    ➥ / 1024);
    var prozent = Math.ceil ((this.getBytesLoaded ()
    ➥ / this.getBytesTotal ()) * 100);
    this._parent.counter_txt.text = "Lade " +
    ➥ prozent + "% of " + kilobytes + "k";
    this._parent.loading_txt.text =
    ➥ "Bild wird geladen - Bitte warten!";
    this._parent.status_mc._width = prozent * 2;
    if (prozent == 100)
    {
      i++;
      if (i == 20)
      {
        this.loaded = true;
        this._parent.counter_txt.text = "";
        this._parent.loading_txt.text = "";
        this._parent.status_mc._visible = 0;
        delete i;
      }
    }
  }
}
```

▶ Schritt 6:

Sie können den Film testen und jederzeit weitere Bilder mit Hilfe der *XML*-Datei hinzufügen bzw. bestehende Bilddaten auch wieder entfernen, und dies ohne den Flash-Film erneut publizieren zu müssen.

Abbildung 3.10: Dynamische Bildgalerie im Einsatz

3.1.3 Dynamische Flash-Banner auf Basis von XML

Im folgenden Workshop geht es um die Umsetzung eines dynamischen Flash-Banners, dessen Inhalte sich mit Hilfe einer separaten *XML*-Datei festlegen lassen.

▶ Schritt 1:

Wir benötigen als Erstes einen Flash-Film, diesen sollten Sie unter dem Namen *Dynamic_Bildergalerie.fla* speichern. Die Bühne sollte *640* Pixel breit und *400* Pixel hoch sein.

▶ Schritt 2:

Sie benötigen drei Movieclip-Instanzen mit den Instanznamen `loading_mc`, `fehler_mc` und `fix`. Zusätzlich wird noch ein dynamisches Textfeld mit dem Variablennamen `titel` benötigt. Innerhalb des Movieclips *Fix_Clip* befindet sich eine Instanz `info` des Movieclips *Info_Clip*. Im Movieclip *Info_Clip* befindet sich noch das dynamische Textfeld mit dem Variablennamen `inhalt`. Die Verschachtelung mag etwas gewöhnungsbedürftig sein, aber Flash macht es möglich und im Flash-Banner dienen diese Verschachtelungen der jeweiligen Zuweisung von Inhalten, wie dem Bannertitel (Variable `titel`) und der Bannerbeschreibung (Variable `inhalt`).

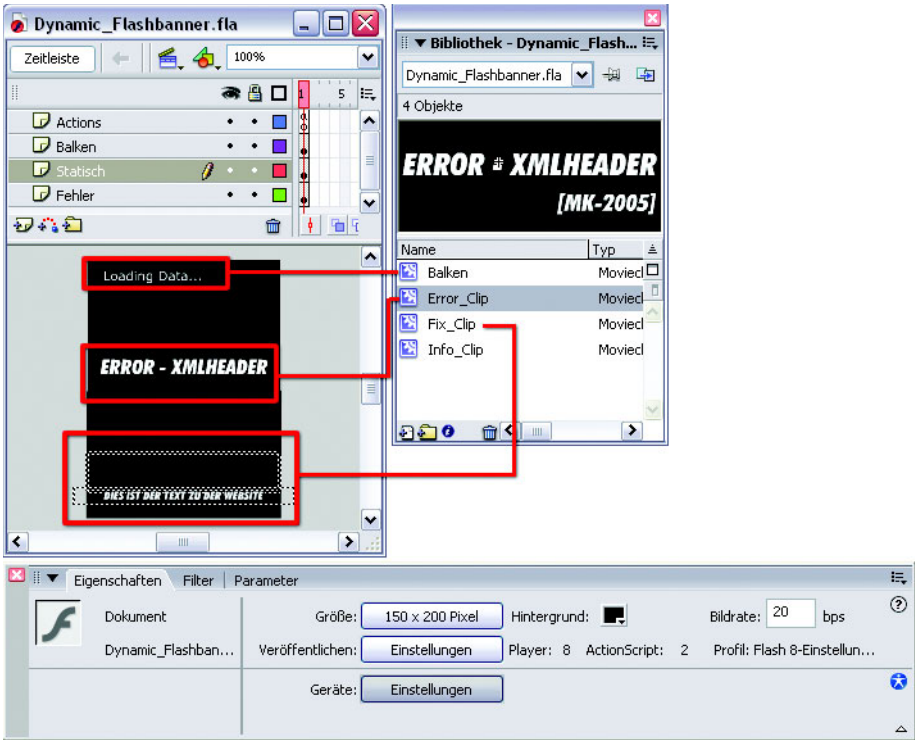


Abbildung 3.11: Aufbau des Flash-Films samt wesentlicher Movieclip-Instanzen

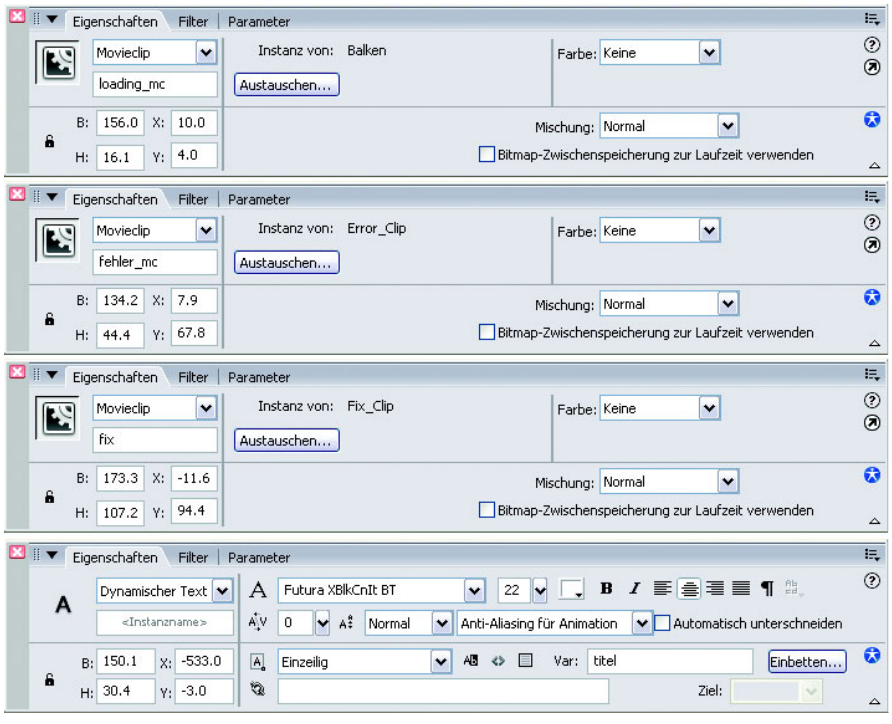


Abbildung 3.12: Movieclip-Instanzen und das dynamische Textfeld mit dem Variablennamen titel

▶ Schritt 3:

Als Nächstes sollten Sie sich um den Inhalt kümmern. Hierfür benötigen Sie eine *XML*-Datei, die die Bannerdaten enthält und unter dem Namen *bannerdaten.xml* gespeichert werden sollte.

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
  <header titel="FLASH:MX 2004" path="bannerbilder/
  ➥ flashhot.jpg" link="http://www.amazon.de/exec
  ➥ /obidos/ASIN/3772379036" inhalt="FLASH:MX 2004
  ➥ - HOTSTUFF" />
  <header titel="FLASH:MX 2004" path="bannerbilder/
  ➥ flashmxpro04.jpg" link="http://www.amazon.de/
  ➥ exec/obidos/ASIN/3772379958" inhalt="FLASH:MX
  ➥ 2004 - PROFESSIONAL" />
  <header titel="ACTION:SCRIPT" path="bannerbilder/
  ➥ aspraxis.jpg" link="http://www.amazon.de/exec/
  ➥ obidos/ASIN/3772367976" inhalt="ACTION:SCRIPT
  ➥ - PRAXISBUCH" />
</root>
```

▶ Schritt 4:

Im ersten Schlüsselbild der Zeitleiste platzieren Sie in der *Aktionen*-Ebene folgende ActionScript-Codezeilen.

```
// Verarbeiten der geladenen Bannerdaten
function ladeHeaders (){
  var anzahl = header.firstChild.childNodes.length;
  var aktHeader = Math.floor (Math.random () *
  ➥ anzahl);
  this.createEmptyMovieClip ("halter", 1);
  loadMovie (header.firstChild.childNodes
  ➥ [aktHeader].attributes.path, "halter");
  this.loading_mc.onEnterFrame = function ()
  {
```

```
    this.gesamt = halter.getBytesTotal ();
    this.bereits = halter.getBytesLoaded ();
    this.prozent = this.bereits * 100 / this.gesamt;
    if (this.prozent == 100)
    {
      removeMovieClip (this);
      delete this.onEnterFrame;
    }
  };
  this.fix.titel = header.firstChild.childNodes
  ➥ [aktHeader].attributes.titel;
  this.fix.info.inhalt = header.firstChild.
  ➥ childNodes[aktHeader].attributes.inhalt;
  this.fix._visible = false;
  this.createEmptyMovieClip ("link", 0);
  with (this.link)
  {
    beginFill (0x000000, 100);
    moveTo (0, 0);
    lineTo (150, 0);
    lineTo (150, 200);
    lineTo (0, 200);
    lineTo (0, 0);
    endFill ();
  }
  this.link.onRollOver = function ()
  {
    _root.fix._visible = true;
  };
  this.link.onRollOut = function ()
  {
    _root.fix._visible = false;
  };
  this.link.onRelease = function ()
  {
    getURL (header.firstChild.childNodes
  ➥ [aktHeader].attributes.link, "_blank");
  };
}
// Laden der Bannerdaten via XML
this.header = new XML ();
this.header.ignoreWhite = true;
this.header.onLoad = function (status){
  if (status)
  {
    ladeHeaders ();
  }
  else
  {
    fehler_mc._visible = 1;
  }
};
this.header.load ("bannerdaten.xml");

this.fehler_mc._visible = 0;
this.fix.swapDepths (2);
this.loading_mc.swapDepths (10);
```

▶ Schritt 5:

Sie können den Film testen und jederzeit weitere Banner mit Hilfe der *XML*-Datei hinzufügen bzw. bestehende Bannerdaten auch wieder entfernen.

Abbildung 3.13: Flash-Banner der besonderen Art – einige Buchcover samt Titel und kurzer Beschreibung

3.1.4 Dynamische Bilder – Duplizieren und Skalieren

Im folgenden Workshop stellen wir Ihnen die `BitmapData`-Klasse und ihre Fähigkeit dynamisch geladene Bilddaten in Echtzeit verarbeiten zu können vor. Die Erzeugung von in Echtzeit generierten Bitmap-Duplikaten bzw. -Klonen wird hiermit unter Einsatz der `draw()`-Methode zum Kinderspiel. Darüber hinaus bietet Ihnen die `Matrix`-Klasse die Möglichkeit der Pixelmanipulation, wie sie z.B. bei der Skalierung von Bildern benötigt wird.

▶ Schritt 1:

Wir benötigen als Erstes einen Flash-Film, diesen sollten Sie unter dem Namen *Bilderklonen_duplizieren_Simple .fla* speichern. Innerhalb der Zeitleiste wurde die *Aktionen*- und *Beschreibung*-Ebene platziert. Zusätzlich werden insgesamt vier Schlüsselbilder benötigt, zwei im ersten Bild und zwei im zweiten Bild der Zeitleiste. Innerhalb der *Beschreibung*-Ebene befinden sich statische Textfelder.

▶ Schritt 2:

Im ersten Schlüsselbild der Zeitleiste platzieren Sie in der *Aktionen*-Ebene folgende ActionScript-Codezeilen.

```
// MovieClipLoader-Ereignis (onLoadComplete)
function onLoadComplete(){
  play();
}

// Erzeugen der Bild-Movieclip-Instanz
var mc:MovieClip =
➥ this.createEmptyMovieClip("mc", 1);

var meinLoader:MovieClipLoader =
➥ new MovieClipLoader();
meinLoader.addListener(_root);
meinLoader.loadClip("spacebild.jpg", mc);

stop();
```

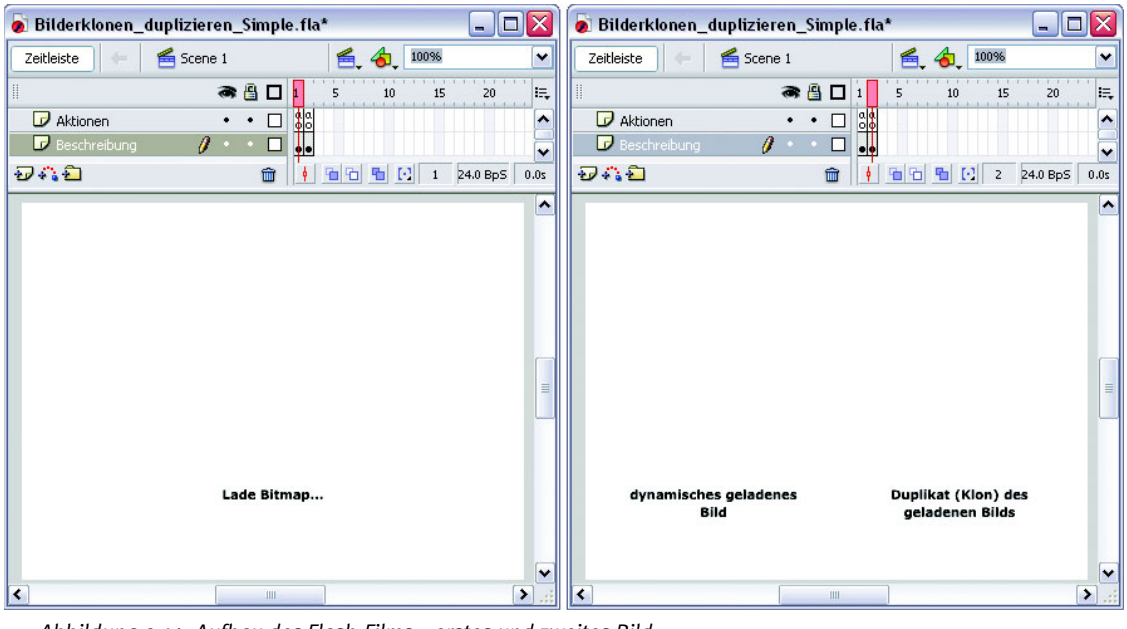


Abbildung 3.14: Aufbau des Flash-Films – erstes und zweites Bild

Schritt 3:

Im zweiten Schlüsselbild der Zeitleiste platzieren Sie in der *Aktionen*-Ebene folgende ActionScript-Codezeilen.

```
// Erzeugen der Duplikat-Movieclip-Instanz
var mc2:MovieClip =
➥ this.createEmptyMovieClip("mc2", 2);
mc2._x = mc._width + 1;
mc2._y = 0;

// Erzeugen der BitmapData-Klassen-Instanz
// mc._width und mc._height (Pixel - Breite x Höhe)
bilddaten = new flash.display.BitmapData(mc._width,
➥ mc._height, true, 0);

// Skalierte Duplikate zeichnen
bilddaten.draw(mc);
mc2.attachBitmap(bilddaten, 1, "auto", true);

stop();
```

Schritt 4:

Sie können den Film testen und sollten selbstverständlich darauf achten, dass sich die Bilddatei *spacebild.jpg* im selben Verzeichnis befindet wie die SWF-Datei. Andernfalls müssten Sie der `loadClip()`-Methode noch einen Pfad übergeben.

Abbildung 3.15: Original und Duplikat (Klon) – das Ergebnis kann sich sehen lassen

Sie werden sicher noch die Skalierung vermissen. Gar kein Problem, hierfür müssen wir lediglich die ActionScript-Codezeilen aus Schritt 3 etwas anpassen. Die Manipulation der Bildpunkte (Pixel) erfolgt mit Hilfe der Transformationsmatrix. Wie wäre es, wenn Sie die bisherigen ActionScript-Codezeilen des zweiten Schlüsselbilds durch folgende ersetzen:

```
// Erzeugen der Duplikat-Movieclip-Instanz
var mc2:MovieClip =
➥ this.createEmptyMovieClip("mc2", 2);
mc2._x = mc._width + 1;
mc2._y = 0;

// Erzeugen der BitmapData-Klassen-Instanz
// mc._width und mc._height (Pixel - Breite x Höhe)
bilddaten = new flash.display.BitmapData(mc._width,
➥ mc._height, true, 0);

// Erzeugen der Transformationsmatrix-Instanz
transmatrix = new flash.geom.Matrix();
// Skalierung um 50 % (Breite und Höhe)
transmatrix.scale(0.5, 0.5);

// Skalierte Duplikate zeichnen
bilddaten.draw(mc, transmatrix);
mc2.attachBitmap(bilddaten, 1, "auto", true);

stop();
```

Abbildung 3.16: Original und skaliertes Duplikat – die Skalierung beträgt genau 50 %

Sollte Sie das immer noch nicht beeindruckt haben, wie wäre es dann mit mehreren Duplikaten, deren Darstellung durch Mischmodus und Filter beeinflusst wird? Hierfür verwenden Sie folgende ActionScript-Codezeilen:

```
// Erzeugen der Duplikat-MovieClip-Instanzen
var mc2:MovieClip =
➥ this.createEmptyMovieClip("mc2", 2);
mc2._x = mc._width + 1;
mc2._y = 0;
var mc3:MovieClip =
➥ this.createEmptyMovieClip("mc3", 3);
mc3._x = mc._width + 102;
mc3._y = 0;
var mc4:MovieClip =
➥ this.createEmptyMovieClip("mc4", 4);
mc4._x = mc._width + 1;
mc4._y = mc._height/2;
var mc5:MovieClip =
➥ this.createEmptyMovieClip("mc5", 5);
mc5._x = mc._width + 102;
mc5._y = mc._height/2;

// Erzeugen der BitmapData-Klassen-Instanz
// 100x100 Pixel (Breite x Höhe)
bilddaten = new flash.display.BitmapData
➥ (mc._width/2, mc._height/2, true, 0);

// Erzeugen der Transformationsmatrix-Instanz
transmatrix = new flash.geom.Matrix();
// Skalierung um 50 % (Breite und Höhe)
transmatrix.scale(0.5, 0.5);

// Skalierte Duplikate zeichnen
bilddaten.draw(mc, transmatrix, new
flash.geom.ColorTransform(), {}, new
flash.geom.Rectangle(0, 0, 200, 200), true);
mc2.attachBitmap(bilddaten, 1, "auto", true);
mc3.attachBitmap(bilddaten, 1, "auto", true);
mc4.attachBitmap(bilddaten, 1, "auto", true);

// Bilddaten klonen
bilddaten2 = bilddaten.clone();

// BlurFilter auf die geklonten Bilddaten anwenden
bilddaten2.applyFilter(bilddaten2, new
flash.geom.Rectangle(0, 0, mc._width/2,
➥ mc._height/2), new flash.geom.Point(0, 0),
➥ new flash.filters.BlurFilter(3, 3, 100));
mc5.attachBitmap(bilddaten2, 1, "auto", true);

// Mischmodi auf mc3 und mc4 anwenden
mc3.blendMode = 14; // hardlight
mc4.blendMode = 7;  // difference

stop();
```

Abbildung 3.17: Original und skalierte Duplikate samt Mischmodus und Filter

Abbildung 3.18: Natürlich können Sie ohne weiteres auch PNG- oder GIF-Bilddaten manipulieren.

3.1.5 Dynamische Videosequenzen und Alpha-Wiedergabe

Im folgenden Workshop stellen wir Ihnen die Einbindung von dynamischen Videosequenzen vor. Sie haben seit Flash 8 die Möglichkeit *FLV*-Dateien samt enthaltenem Alphakanal einzulesen. Den Anwendungsmöglichkeiten sind kaum Grenzen gesetzt.

▶ Schritt 1:

Wir benötigen als Erstes einen Flash-Film, diesen sollten Sie unter dem Namen *Externe_Videosequenzen.fla* speichern. Aus dem Komponenten-Bedienfeld, das Sie mit Hilfe der Option Fenster/Komponenten erreichen, ziehen Sie eine Instanz der TextArea-Komponente auf die Bühne. Zusätzlich benötigen Sie eine Movieclip-Instanz (*VideoClip*), in der Sie eine Video-Instanz (*Videocontainer*) platzieren.

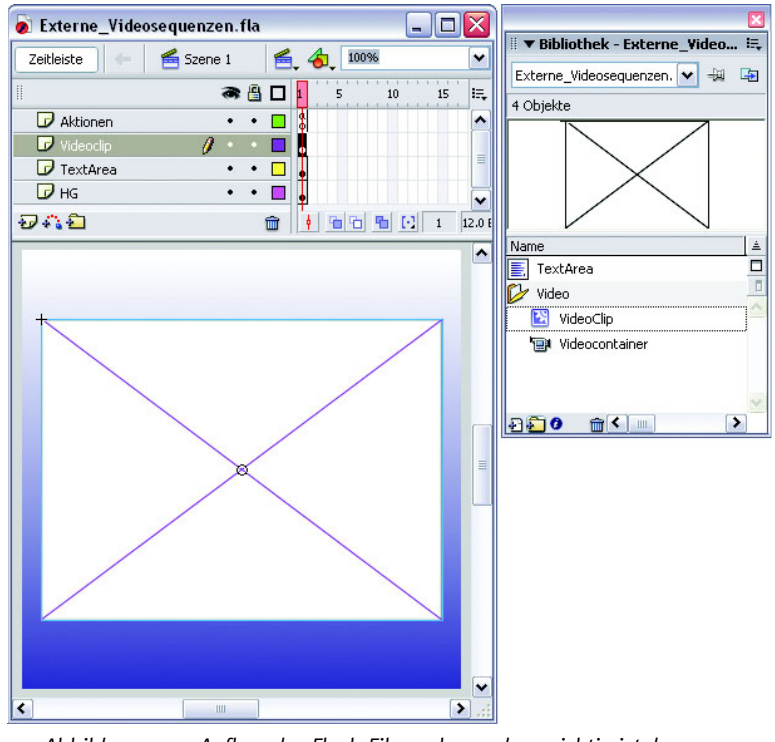


Abbildung 3.19: Aufbau des Flash-Films – besonders wichtig ist der Videocontainer zur Darstellung der FLV-Datei

▶ Schritt 2:

Wählen Sie die TextArea-Komponente und weisen Sie ihr den Instanznamen `display_txt` zu. Als Nächstes weisen Sie der Movieclip-Instanz den Namen `video_mc` zu. Damit die Video-Instanz, die sich innerhalb des *Video-Clip*-Movieclips befindet, angesprochen werden kann, erhält diese den Instanznamen `video`.

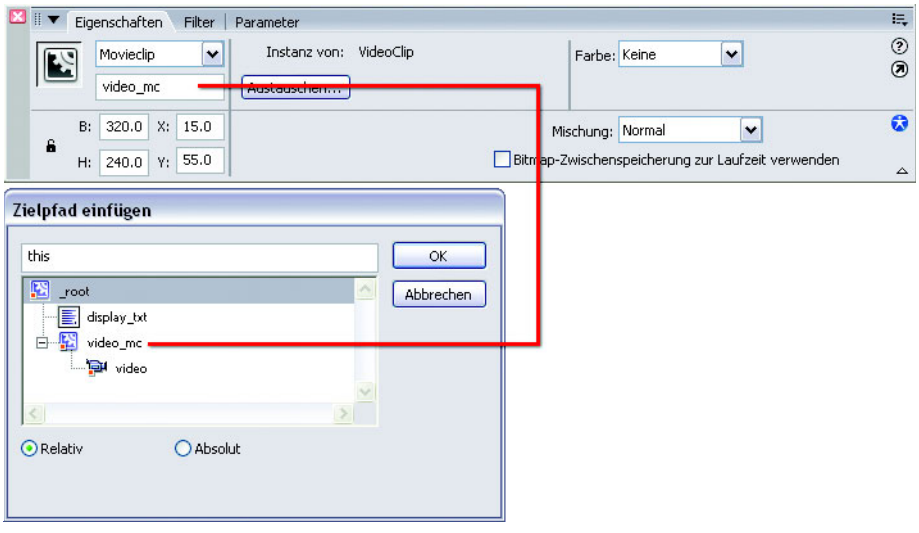


Abbildung 3.20: Instanznamen und Verschachtelung der wesentlichen Bestandteile

▶ Schritt 3:

Im ersten Schlüsselbild der Zeitleiste platzieren Sie in der *Aktionen*-Ebene folgende ActionScript-Codezeilen.

```
// Verbindung
var verbindung = new NetConnection ();
verbindung.connect (null);

// Stream samt Endlosschleife
var stream = new NetStream (verbindung);
video_mc.video.attachVideo (stream);
stream.play ("shine.flv");
stream.onStatus = function (info){
  if (info.code == "NetStream.Play.Stop")
  {
    stream.seek (0);
  }
};

display_txt.text = "Platzhalter Text...";
display_txt.setStyle ("backgroundColor", 0x0A2249);
display_txt.setStyle ("color", 0xDDEBFB);
```

Schritt 4:

Sie können den Film testen und wie Sie sehen, lässt sich die *FLV*-Datei durch den integrierten Alphakanal direkt über den Inhalt der TextArea-Komponenten-Instanz platzieren und eröffnet Ihnen neue Gestaltungsmöglichkeiten.

Abbildung 3.21: Videoinhalte direkt über Textinhalten platziert, dies stellt kein Problem dar

3.1.6 Dynamische Videosequenzen und Schaltflächen

Im folgenden Workshop stellen wir Ihnen die Einbindung von dynamischen Videosequenzen vor, die durch ein Nutzerereignis ausgelöst werden. In diesem Zusammenhang darf der Einsatz von Schaltflächen natürlich nicht zu kurz kommen. An dieser Stelle sollten wir noch darauf hinweisen, dass die innerhalb der *FLV*-Datei enthaltenen Videoeffekte mit *Adobe After Effects* realisiert wurden. Über *Adobe After Effects* wird auch der Alphakanal realisiert, so dass eine fehlerfreie Wiedergabe sichergestellt werden kann.

▶ Schritt 1:

Wir benötigen als Erstes einen Flash-Film, diesen sollten Sie unter dem Namen *Video_Schalter_1.fla* speichern. Aus dem Komponenten-Bedienfeld, das Sie mit Hilfe der Option Fenster/Komponenten erreichen, ziehen Sie jeweils eine Instanz der NumericStepper- und der TextArea-Komponente auf die Bühne. Zusätzlich benötigen Sie eine MovieClip-Instanz (*VideoClip*), in der Sie eine Video-Instanz (*Videocontainer*) platzieren.

▶ Schritt 2:

Wählen Sie die TextArea-Komponente und weisen Sie ihr den Instanznamen `display_txt` zu. Als Nächstes weisen Sie der NummericStepper-Komponente den Namen `stepper` und den drei Schalter_Video_Clip-Instanzen die Instanznamen `b1` bis `b3` zu. Innerhalb des Schalter_Video_Clip-Symbols befinden sich die Schalter-Instanz `roll_btn` und die MovieClip-Instanz `video_mc`. Damit die Video-Instanz, die sich innerhalb des VideoClip-Movieclips befindet, angesprochen werden kann, erhält diese den Instanznamen `video`. Bitte beachten Sie die Parameter `minimum` und `maximum` der NummericStepper-Komponente, der Parameter `minimum` erhält den Wert *1* und der Parameter `maximum` den Wert *14*.

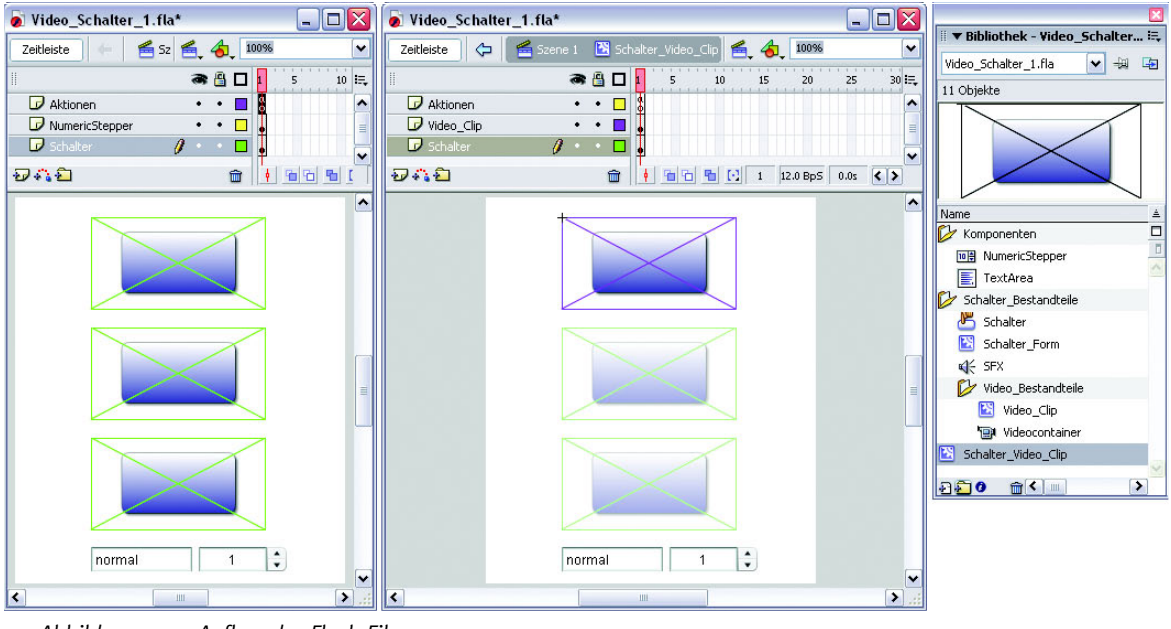


Abbildung 3.22: Aufbau des Flash-Films

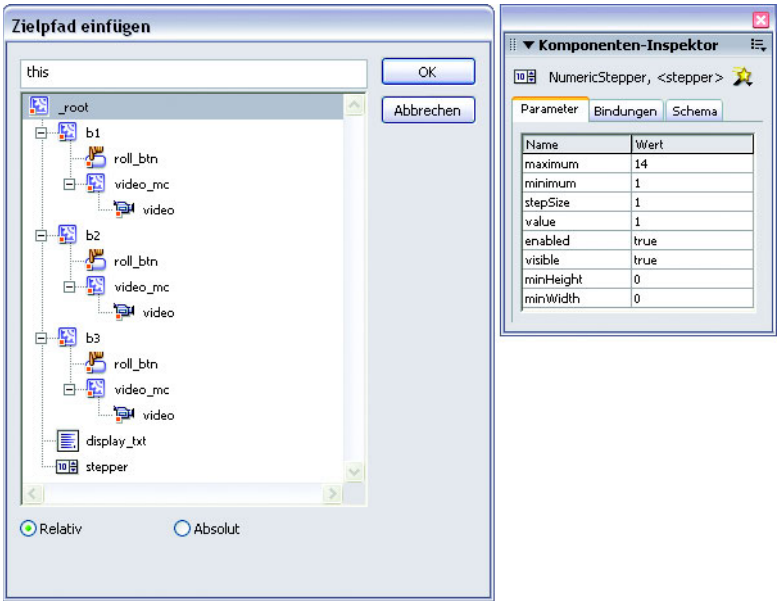


Abbildung 3.23: Verschachtelung der einzelnen Bestandteile und die entsprechenden Instanznamen

▶ Schritt 3:

Im ersten Schlüsselbild der Hauptzeitleiste platzieren Sie in der *Aktionen*-Ebene folgende ActionScript-Codezeilen.

```
var verbindung = new NetConnection ();
verbindung.connect (null);

var modi = new Array ("",
➥ "normal",
➥ "layer",
➥ "multiply",
➥ "screen",
➥ "lighten",
➥ "darken",
➥ "difference",
➥ "add",
➥ "subtract",
➥ "invert",
➥ "alpha",
➥ "erase",
➥ "overlay",
➥ "hardlight");

auswahlListener = new Object ();
auswahlListener.change = function (eventObjekt){
  b1.video_mc.blendMode = eventObjekt.target.value;
  b2.video_mc.blendMode = eventObjekt.target.value;
  b3.video_mc.blendMode = eventObjekt.target.value;
  display_txt.text =
  ➥ modi[eventObjekt.target.value];
};
stepper.addEventListener ("change",
➥ auswahlListener);
```

▶ Schritt 4:

Im ersten Schlüsselbild der Zeitleiste des Schalter_Video_Clip-Symbols platzieren Sie in der *Aktionen*-Ebene folgende ActionScript-Codezeilen.

```
var stream =
➥ new NetStream (this._parent.verbindung);
video_mc.video.attachVideo (stream);
stream.play ("sterne.flv");
stream.pause (true);
stream.onStatus = function (info){
  if (info.code == "NetStream.Play.Stop")
  {
    video_mc._visible = false;
  }
};

roll_btn.onRollOver = function (){
  video_mc._visible = true;
  stream.seek (0);
  stream.pause (false);
};
```

▶ Schritt 5:

Sie können den Film testen und wie man sieht, steht der Realisierung von Video-Schalflächen nun nichts mehr im Wege und die Kombination mit einem Mischmodus liefert oftmals beeindruckende Effekte.

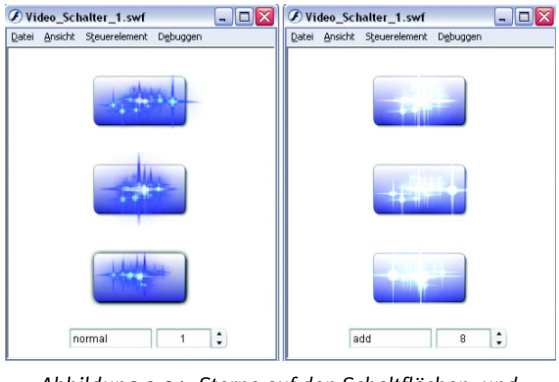


Abbildung 3.24: Sterne auf den Schaltflächen, und dies ganz ohne Hintergrund

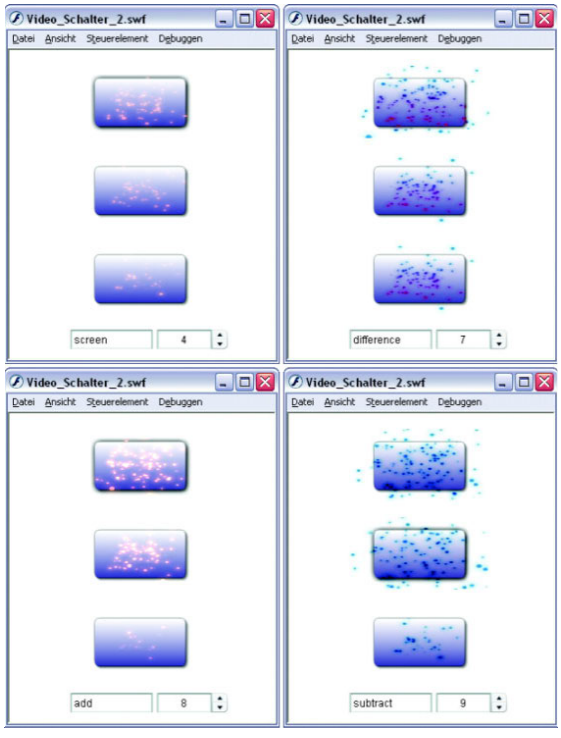


Abbildung 3.25: Wie wäre es mit kleinen Kügelchen? Einfach die FLV-Datei wechseln und schon hat man einen neuen Effekt.

3.1.7 Dynamische Videos – sequenzielle Wiedergabe

In diesem Workshop stellen wir Ihnen vor, wie einfach es ist, eine sequenzielle Videowiedergabe mit Hilfe der FLVPlayback-Komponente zu realisieren. Bei der sequenziellen Wiedergabe wird eine Reihe von einzelnen FLV-Dateien abgerufen und nacheinander abgespielt. Diese Art der Wiedergabe eignet sich vor allem bei Videotrainings bzw. bei Videoinhalten die auf mehrere einzelne Videosequenzen verteilt vorliegen.

Schritt 1:

Wir benötigen als Erstes einen Flash-Film, diesen sollten Sie unter dem Namen *Mehrere_Videos_Abspielen.fla* speichern. Aus dem Komponenten-Bedienfeld, das Sie mit Hilfe der Option Fenster/Komponenten erreichen, ziehen Sie eine Instanz der FLVPlayback-Komponente auf die Bühne und weisen dieser den Instanznamen `mein_controller` zu. Zusätzlich benötigen Sie eine dynamische Textfeld-Instanz, die den Instanznamen `display_txt` erhält.

Schritt 2:

Sie sollten dem FLVPlayback-Parameter `contentPath` die erste *FLV*-Datei zuweisen, so dass die Wiedergabe der Videos gewährleistet wird. Es steht Ihnen darüber hinaus frei, auch noch ein FLVPlayback-Skin auszuwählen und der FLVPlayback-Komponente zuzuweisen. Hierfür müssen Sie lediglich den FLVPlayback-Parameter `skin` bearbeiten.

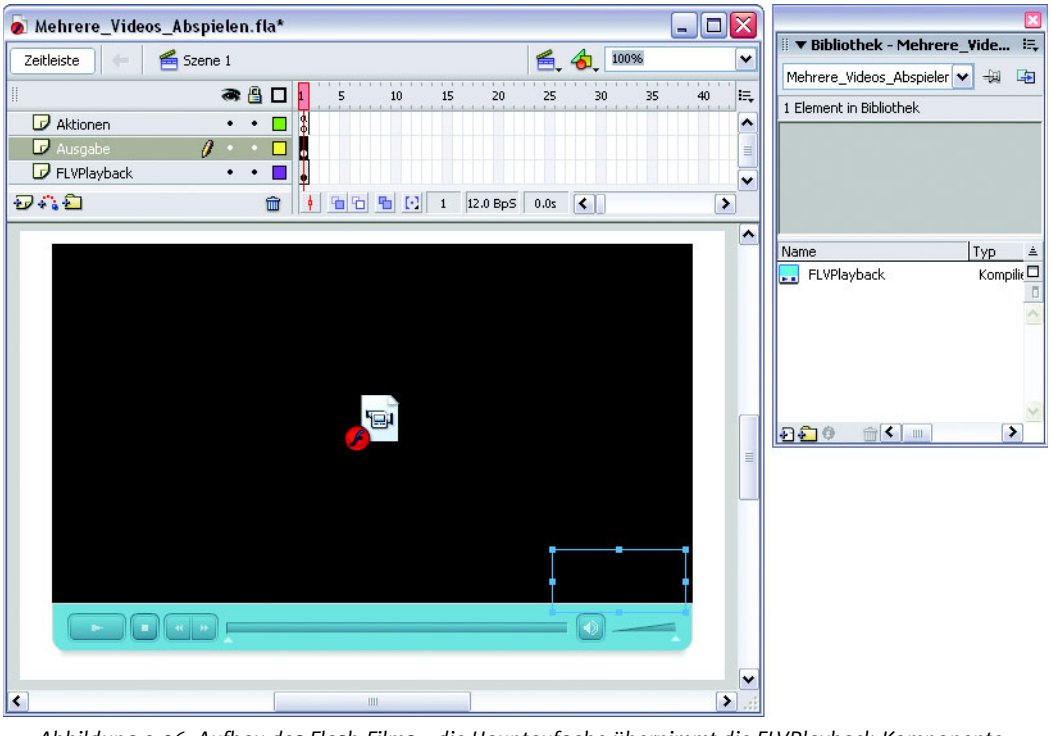


Abbildung 3.26: Aufbau des Flash-Films – die Hauptaufgabe übernimmt die FLVPlayback-Komponente

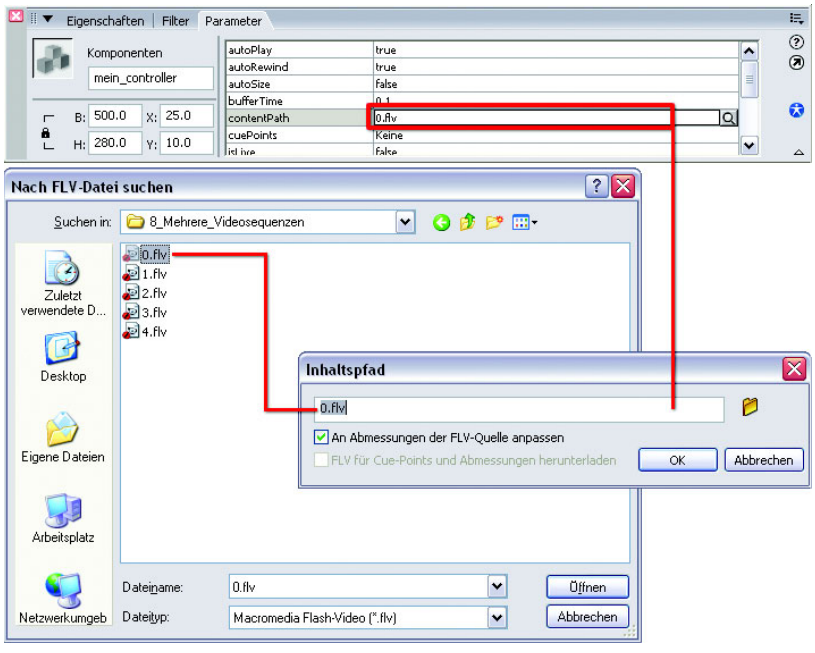


Abbildung 3.27: Sie sollten der FLVPlayback-Komponente die erste FLV-Datei zuweisen.

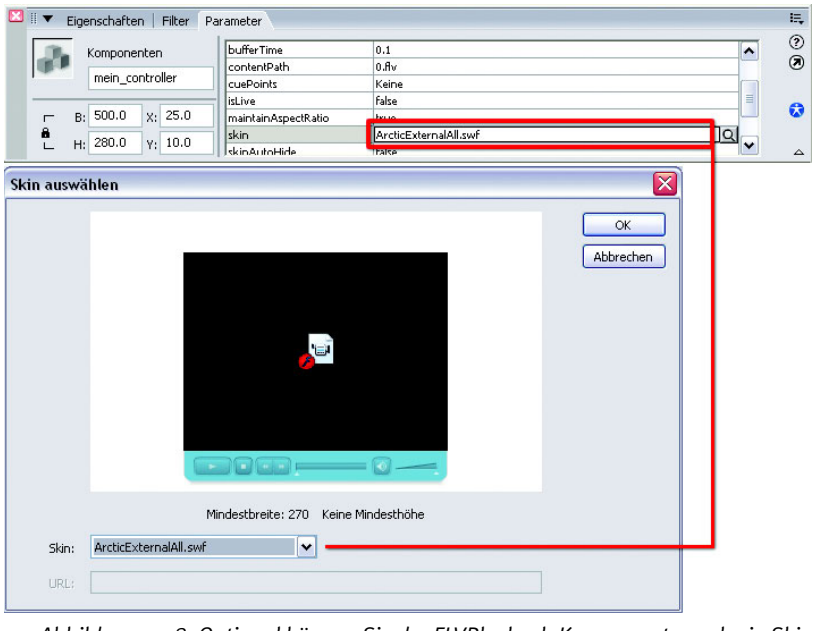


Abbildung 3.28: Optional können Sie der FLVPlayback-Komponente auch ein Skin zuweisen.

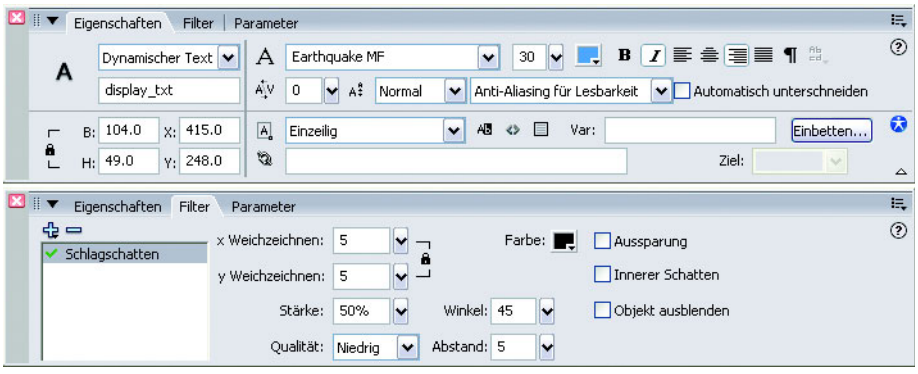


Abbildung 3.29: Dem dynamischen Textfeld display_txt wurde der Filter Schlagschatten zugewiesen, dies stellt lediglich eine optische Anpassung dar.

▶ Schritt 3:

Im ersten Schlüsselbild der Hauptzeitleiste platzieren Sie in der *Aktionen*-Ebene folgende ActionScript-Codezeilen.

```
// Initialisierung der Ausgangswerte
var anzahlFLV:Number = 4;
var readyIndex:Number = 0;
var videoIndex:Number = 0;
mein_controller.contentPath = videoIndex+".flv"
display_txt.text = mein_controller.contentPath;

// Die Ereignismethode spieleFLV sorgt dafür, dass
// jeweils die nächste FLV-Datei abgerufen wird
function spieleFLV(e:Object):Void {
  // Weiter zur nächsten FLV-Datei bis zur letzten
  if (readyIndex<anzahlFLV) {
    readyIndex++;
    mein_controller.activeVideoPlayerIndex =
    ➥ readyIndex;
    mein_controller.load( readyIndex+".flv" );
  }
}
mein_controller.addEventListener("ready",
➥ spieleFLV);
```

```
// Die Ereignismethode getFinal spielt das jeweils
// nächste FLV-Video ab, sobald das Ende der
// vorherigen Videosequenz erreicht wurde
function getFinal(e:Object):Void {
  videoIndex =
  ➥ (videoIndex<anzahlFLV) ? ++videoIndex: 0;
  mein_controller.activeVideoPlayerIndex =
  ➥ videoIndex;
  mein_controller.visibleVideoPlayerIndex =
  ➥ videoIndex;
  mein_controller.play();
  display_txt.text = e.target.contentPath;
}
mein_controller.addEventListener("complete",
➥ getFinal);
```

▶ Schritt 4:

Sie können den Film testen. Wie Sie feststellen werden, werden die *FLV*-Dateien nacheinander wiedergegeben und der sequenziellen Wiedergabe steht nichts mehr im Weg.

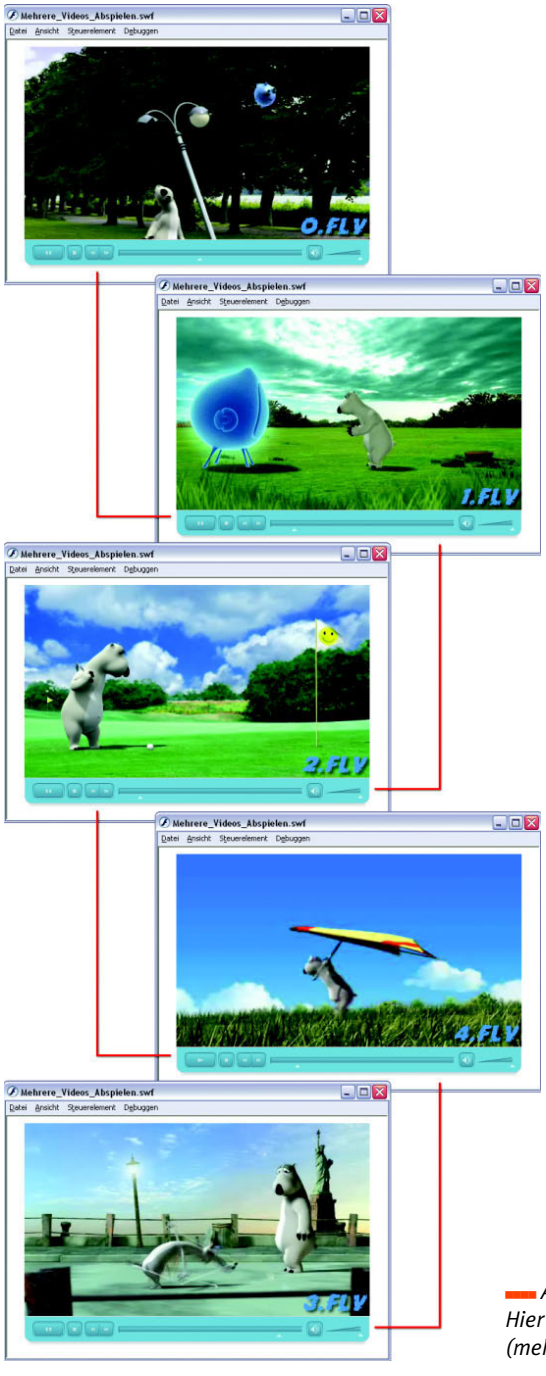


Abbildung 3.30:
Hier tanzt der Eisbär
(mehr davon auf der Buch-CD).

3.1.8 Dynamische Videos und Übergänge (Transitions)

In diesem Workshop stellen wir Ihnen vor, wie einfach es ist, einen Übergang zwischen Videosequenzen zu realisieren. Ganz nebenbei erfahren Sie auch noch, dass Flash in der Lage ist, mehr als eine Videosequenz unter Kontrolle zu halten und synchron zu verwalten.

▶ Schritt 1:

Wir benötigen als Erstes einen Flash-Film, diesen sollten Sie unter dem Namen *Video_Uebergaenge.fla* speichern. Aus dem Komponenten-Bedienfeld, das Sie mit Hilfe der Option Fenster/Komponenten erreichen, ziehen Sie eine Instanz der FLVPlayback-Komponente auf die Bühne und weisen dieser den Instanznamen `mein_controller` zu. Zusätzlich benötigen Sie vier Schaltflächen-Instanzen, deren Instanznamen `b1_btn` bis `b4_btn` lauten. Diese werden Sie in die Lage versetzen, zwischen den Videosequenzen zu wechseln und den Übergangseffekt auszulösen.

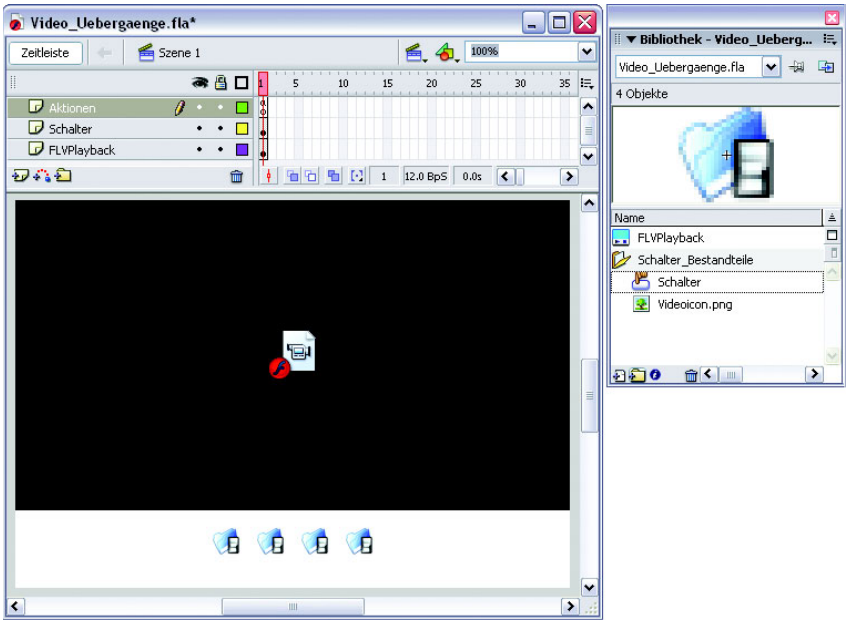


Abbildung 3.31: Aufbau des Flash-Films – die Hauptrolle spielt auch in diesem Fall die FLVPlayback-Komponente

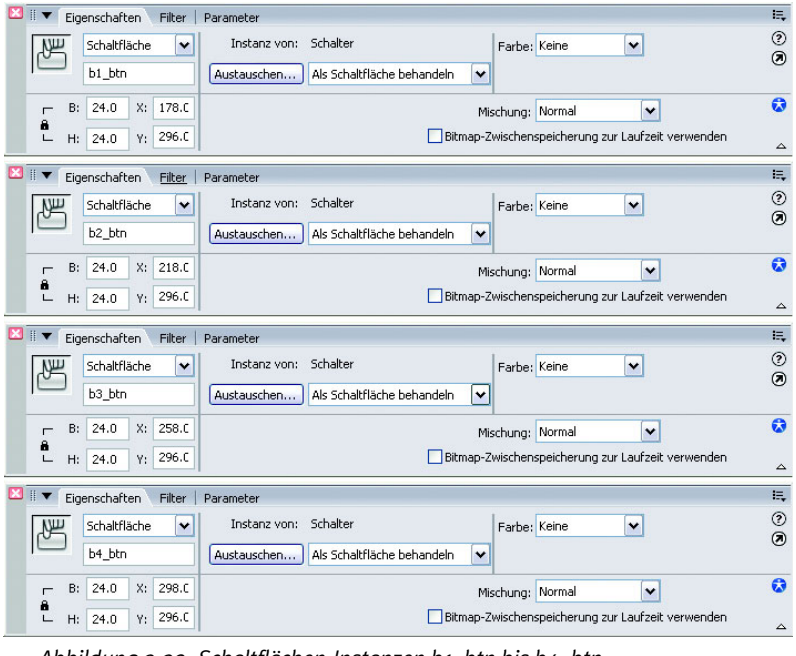


Abbildung 3.32: Schaltflächen-Instanzen b1_btn bis b4_btn

Schritt 2:

Sie sollten dem FLVPlayback-Parameter `contentPath` die erste *FLV*-Datei zuweisen, so dass die Wiedergabe gewährleistet wird.

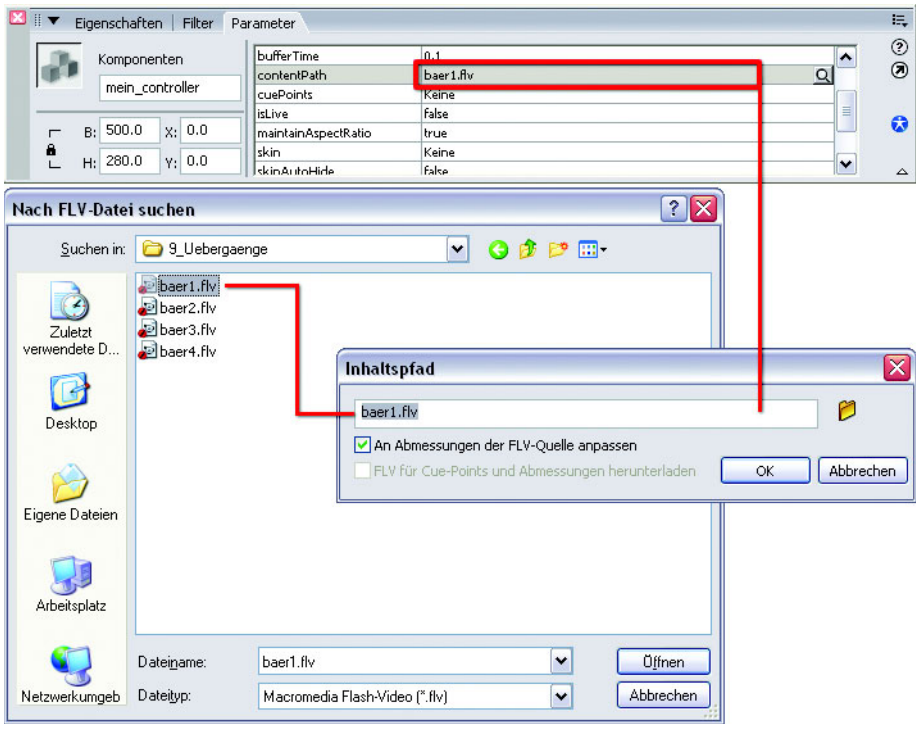


Abbildung 3.33: Sie sollten der FLVPlayback-Komponente die erste FLV-Datei zuweisen.

Schritt 3:

Im ersten Schlüsselbild der Hauptzeitleiste platzieren Sie in der *Aktionen*-Ebene folgende ActionScript-Codezeilen.

```
import mx.transitions.*;
var anzahlFLV:Number = 4;
var readyIndex:Number = 0;

// Sämtliche FLV-Dateien in die FLVPlayback-
// Komponente laden
// Insgesamt entstehen vier einzelne Videoplayer
for (var i =1; i<=anzahlFLV; i++) {
  mein_controller.activeVideoPlayerIndex = i;
  mein_controller.load("baer"+i+".flv");
}
```

```
// Die Ereignismethode spieleFLV startet die
// jeweils geladene FLV-Datei
function spieleFLV(eventObjekt:Object) {
  readyIndex++;

  // Insgesamt werden vier Videos geladen
  // sobald der readyIndex der Anzahl
  // der FLV-Dateien entspricht, kann es mit der
  // Wiedergabe beginnen
  if (readyIndex == anzahlFLV) {
    for (var i = 1; i<=anzahlFLV; i++) {
      // Legt den aktuellen Videoplayer fest und
      // spielt das Video ab
      eventObjekt.target.activeVideoPlayerIndex = i;
      eventObjekt.target.play();
    }
  }
}
mein_controller.addEventListener("ready",
➥ spieleFLV);

// Transition (Übergang)
function transDone(eventObjekt:Object) {
  mein_controller.visibleVideoPlayerIndex =
  ➥ eventObjekt.target.content._name;
}

function setzeTransition(mc:MovieClip, nr:Number,
➥ dauer:Number) {
    if (nr != mc.visibleVideoPlayerIndex) {
      var videoplayer:MovieClip =
      ➥ mc.getVideoPlayer(nr);
      mc.bringVideoPlayerToFront(nr);
      // Blenden-Übergang
      TransitionManager.start(videoplayer,
      ➥ {type:mx.transitions.Fade,
      ➥ direction:0,
      ➥ duration:dauer,
      ➥ easing:mx.transitions.easing.None.easeNone,
      ➥ param1:empty,
      ➥ param2:empty});
      videoplayer.__transitionManager.
      ➥ addEventListener("allTransitionsInDone",
      ➥ transDone);
    }
}

// Videoschleife (wiederholt die Wiedergabe,
// wenn das Video abgelaufen ist)
function getFinal(eventObjekt:Object) {
  for (var i = 1; i<=anzahlFLV; i++) {
    eventObjekt.target.activeVideoPlayerIndex = i;
    eventObjekt.target.play();
  }
}
mein_controller.addEventListener("complete",
➥ getFinal);

// Filter und Ereignisse für die Schalter
function filterEin(obj:MovieClip) {
  schattenFilter = new flash.filters.
  ➥ DropShadowFilter(0, 0, 0x1B91B9, 3, 25, 25,
  ➥ 1, 3, false, false);
  obj.filters = [schattenFilter];
}
function filterPress(obj:MovieClip) {
  schattenFilter = new flash.filters.
  ➥ DropShadowFilter(0, 0, 0x9D89D9, 3, 25, 25,
  ➥ 1, 3, false, false);
  obj.filters = [schattenFilter];
}
function filterAus(obj:MovieClip) {
  obj.filters = [];
}
function verkleinern(obj:MovieClip) {
  obj._width = 24;
  obj._height = 24;
}
function vergroessern(obj:MovieClip) {
  obj._width = 30;
  obj._height = 30;
}
```

```
b1_btn.onRollOver = function() {
  filterEin(this);
};
b1_btn.onRollOut = function() {
  filterAus(this);
};
b1_btn.onPress = function() {
  setzeTransition(_level0.mein_controller,1,1.0);
  filterPress(this);
  vergroessern(this);
};
b1_btn.onRelease = function() {
  filterEin(this);
  verkleinern(this);
};
b2_btn.onRollOver = function() {
  filterEin(this);
};
b2_btn.onRollOut = function() {
  filterAus(this);
};
b2_btn.onPress = function() {
  setzeTransition(_level0.mein_controller,2,1.0);
  filterPress(this);
  vergroessern(this);
};
b2_btn.onRelease = function() {
  filterEin(this);
  verkleinern(this);
};
b3_btn.onRollOver = function() {
  filterEin(this);
};
b3_btn.onRollOut = function() {
  filterAus(this);
};
b3_btn.onPress = function() {
  setzeTransition(_level0.mein_controller,3,1.0);
  filterPress(this);
  vergroessern(this);
};
b3_btn.onRelease = function() {
  filterEin(this);
  verkleinern(this);
};
b4_btn.onRollOver = function() {
  filterEin(this);
};
b4_btn.onRollOut = function() {
  filterAus(this);
};
b4_btn.onPress = function() {
  setzeTransition(_level0.mein_controller,4,1.0);
  filterPress(this);
  vergroessern(this);
};
b4_btn.onRelease = function() {
  filterEin(this);
  verkleinern(this);
};
```

▶ **Schritt 4:**

Sie können den Film testen. Wie Sie feststellen werden, werden die *FLV*-Dateien vom Flash Player synchron verwaltet, so dass ein fehlerfreies Ausführen des Übergangseffekts gewährleistet werden kann.

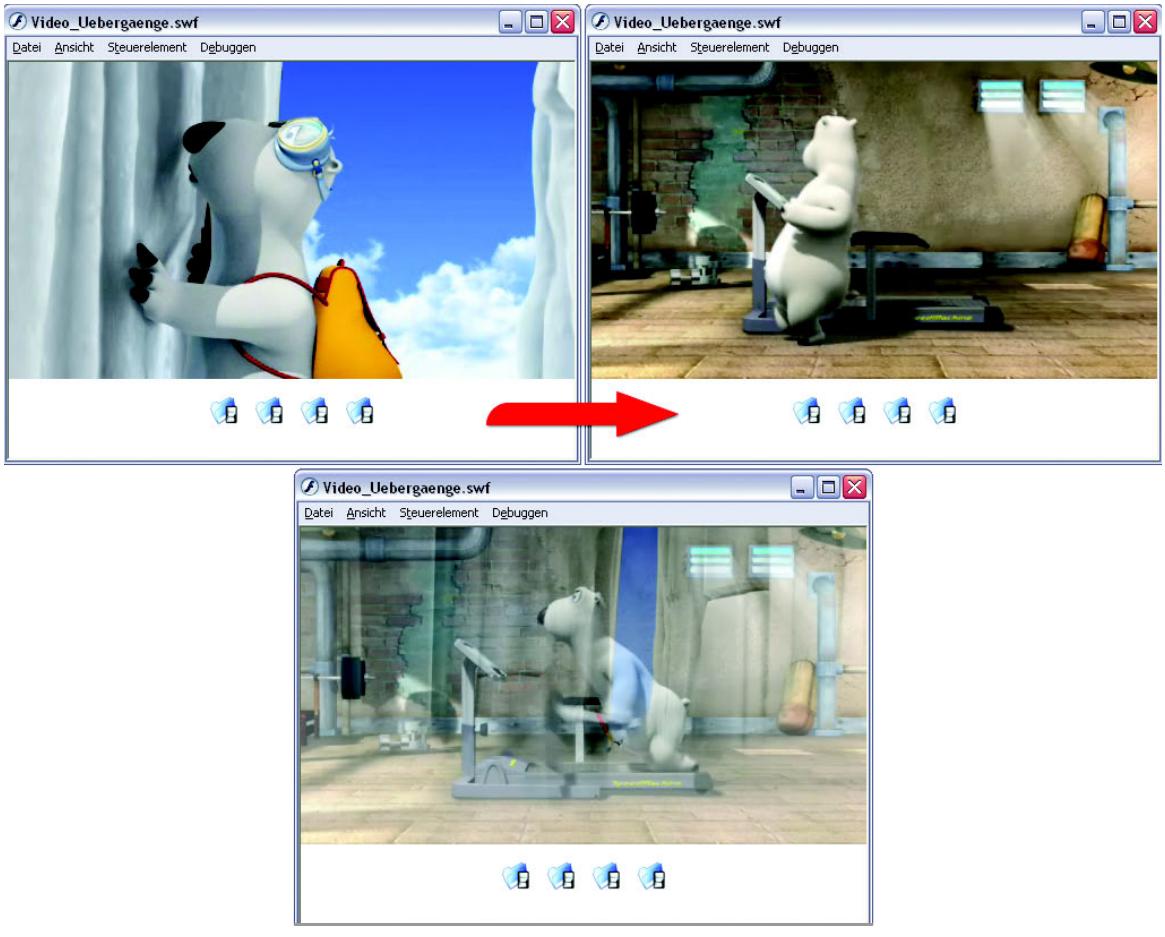


Abbildung 3.34: Übergänge stellen dank der TransitionManager-Klasse kein Problem dar.

3.1.9 Dynamische Videos – Farbe oder Schwarz-Weiß

In diesem Workshop stellen wir Ihnen vor, wie man aus einem Farbbild einer Videosequenz einen Klassiker zaubert, und zwar in Schwarz-Weiß und Hellbraun.

Schritt 1:

Wir benötigen als Erstes einen Flash-Film, diesen sollten Sie unter dem Namen *Video_und_Filter.fla* speichern. Aus dem Komponenten-Bedienfeld, das Sie mit Hilfe der Option Fenster/Komponenten erreichen, ziehen Sie eine Instanz der FLVPlayback-Komponente auf die Bühne und weisen dieser den Instanznamen `mein_controller` zu. Zusätzlich benötigen Sie zwei Schaltflächen-Instanzen, einer weisen Sie den Instanznamen `farben_btn` und der anderen `grau_btn` zu.

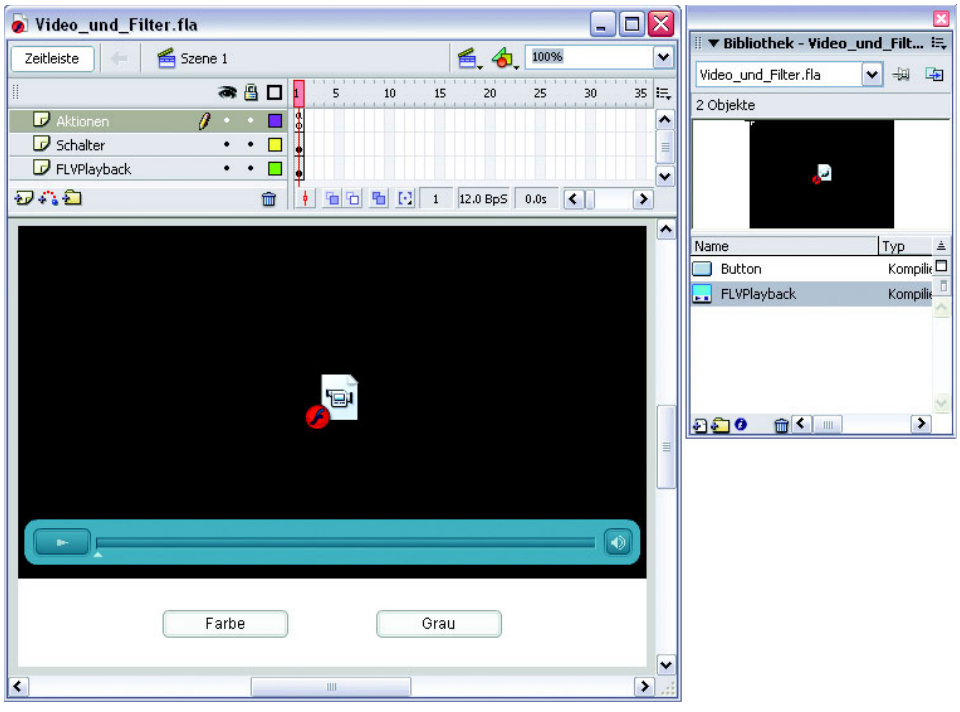


Abbildung 3.35: Aufbau des Flash-Films – die Hauptrolle spielt auch in diesem Fall die FLVPlayback-Komponente

Abbildung 3.36: Schaltflächen-Instanzen farben_btn und grau_btn

▶ Schritt 2:

Sie sollten dem FLVPlayback-Parameter `contentPath` die *FLV*-Datei zuweisen, so dass die Wiedergabe des Videos gewährleistet wird. Es steht Ihnen darüber hinaus frei, auch noch ein FLVPlayback-Skin auszuwählen und der FLVPlayback-Komponente zuzuweisen. Hierfür müssen Sie lediglich den FLVPlayback-Parameter `skin` bearbeiten.

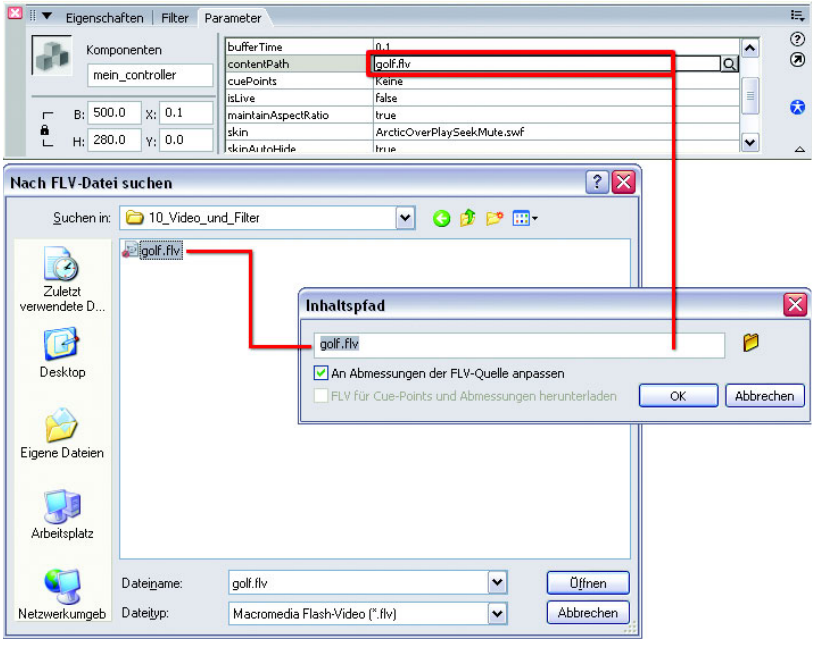


Abbildung 3.37: Sie sollten der FLVPlayback-Komponente die FLV-Datei zuweisen.

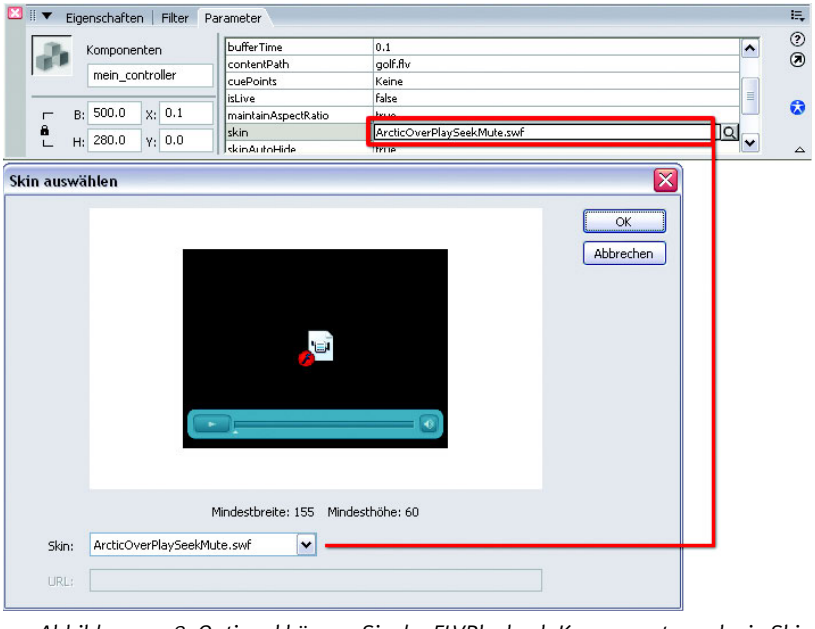


Abbildung 3.38: Optional können Sie der FLVPlayback-Komponente auch ein Skin zuweisen.

Schritt 3:

Im ersten Schlüsselbild der Hauptzeitleiste platzieren Sie in der *Aktionen*-Ebene folgende ActionScript-Codezeilen.

```
// ColorMatrixFilter-Liste, um den Schwarz-Weiss-
// Effekt (Grau) zu realisieren
var intArray:Array = [0.33,0.33,0.33,0,0.33,
➥ 0.33,0.33,0.33,0,0.33,
➥ 0.33,0.33,0.33,0,0.33,
➥ 1,1,1,1,0.33];
// Zuweisen der Anwenden der ColorMatrixFilter-
// Lister
var colorMatrixFilt:Object =
➥ new flash.filters.ColorMatrixFilter (intArray);

// Die Farbwechsel-Methode prüft, welcher Schalter
// gedrückt wurde und weist den entsprechenden
// Filter zu
function farbWechsel(eventObjekt:Object) {
  if (eventObjekt.target == _level0.farben_btn) {
    mein_controller.filters = [];
  } else if (eventObjekt.target ==
  ➥ _level0.grau_btn) {
    mein_controller.filters = [colorMatrixFilt];
  }

}

farben_btn.addEventListener("click", farbWechsel);
grau_btn.addEventListener("click", farbWechsel);
```

Schritt 4:

Sie können den Film testen. Wie Sie feststellen werden, wird die Videosequenz in Farbe wiedergegeben. Sobald Sie jedoch auf die Schaltfläche *Grau* klicken haben Sie ein Schwarz-Weiß-Bild.

Abbildung 3.39: Farbbild

Abbildung 3.40: Schwarz-Weiß (Grau)

Wer nun noch den Klassiker realisieren möchte, sollte einfach die `ColorMatrixFilter`-Liste `initArray` mit folgenden Werten versorgen.

```
// ColorMatrixFilter-Liste, um den Klassik-Effekt
(Hellbraun) zu realisieren
var intArray:Array = [0.60,0.60,0.60,0,1,
          0.20,0.50,0.50,0,1,
          0.20,0.20,0.20,0,1,
          1,1,1,1,1];
```

Abbildung 3.41: Klassiker (Hellbraun)

3.1.10 Upload von Dateien via Flash & PHP

In diesem Workshop geht es darum, Ihnen die neue Upload-Möglichkeit vorzustellen. Sie werden sehen das Ihnen hierbei die `FileReference`-Klasse gute Dienste erweisen wird.

Schritt 1:

Wir benötigen als Erstes einen Flash-Film, diesen sollten Sie unter dem Namen *upload.fla* speichern. Aus dem Komponenten-Bedienfeld, das Sie mit Hilfe der Option Fenster/Komponenten erreichen, ziehen Sie jeweils eine Label-, TextArea und TextInput-Komponente auf die Bühne und weisen der TextArea- und der TextInput-Komponente die Instanznamen `display_txt` und `datei_txt` zu. Zusätzlich benötigen Sie zwei Button-Komponenten, der einen weisen Sie den Instanznamen `browse_btn` und der anderen `upload_btn` zu.

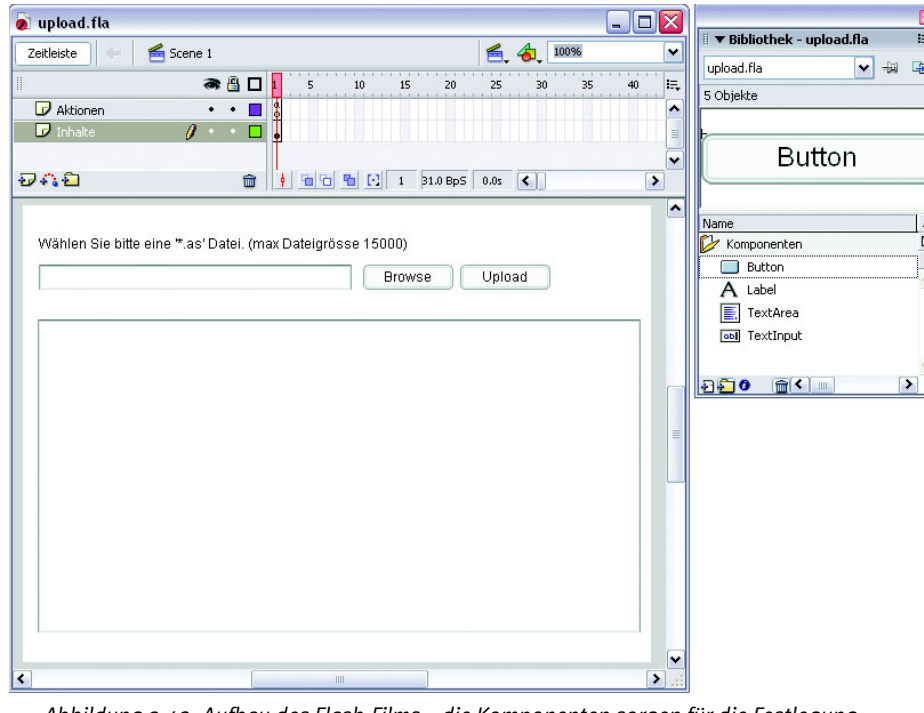


Abbildung 3.42: Aufbau des Flash-Films – die Komponenten sorgen für die Festlegung wichtiger Upload-Optionen

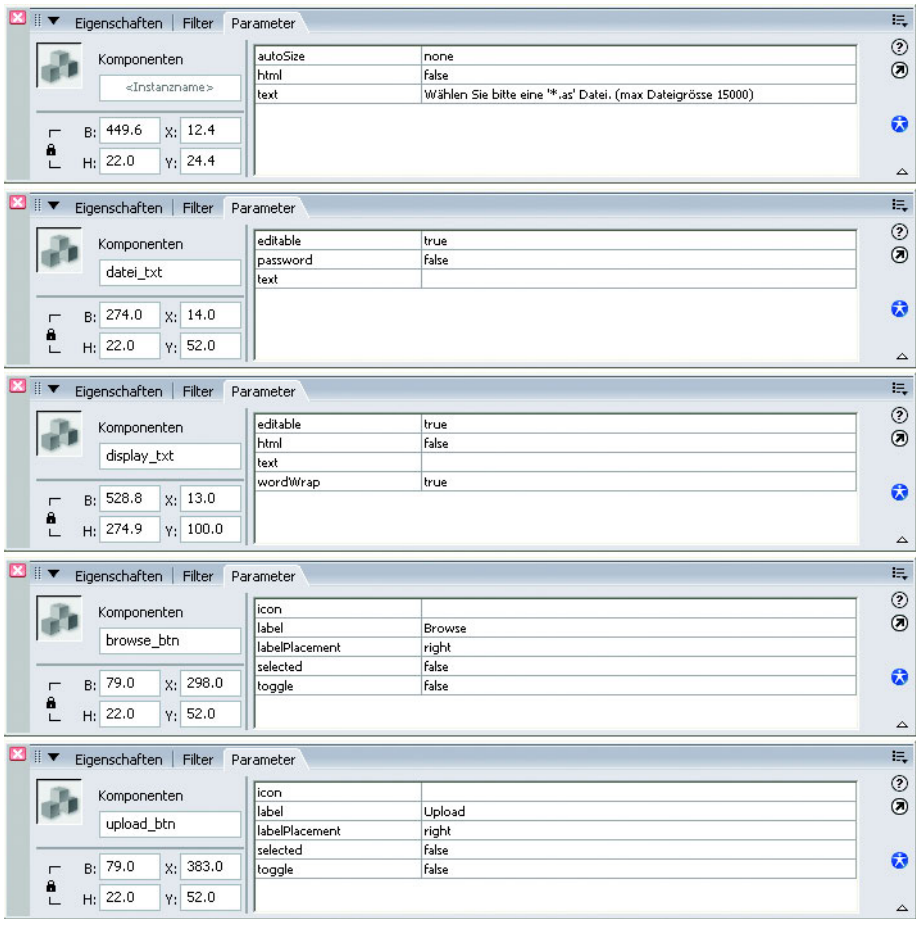


Abbildung 3.43: Sämtliche beteiligten Komponenten und deren Instanznamen

▶ **Schritt 2:**

Im ersten Schlüsselbild der Hauptzeitleiste platzieren Sie in der *Aktionen*-Ebene folgende ActionScript-Codezeilen.

```
// Display (aktualisieren)
updatePosition = function () {
  // Abbruchposition
  display_txt.vPosition = display_txt.maxVPosition;
}
display_txt.addEventListener ("change",
➥ updatePosition);

// Dateiauswahl-Listener
browseListener = new Object();
browseListener.onSelect = function (datei){
```

```
  // Lediglich AS-Dateien und maximal 15000 Bytes
  if (datei.type == ".as" && datei.size <= 15000) {
    display_txt.text = "";
    for (var i in datei){
      display_txt.text += i + ": " + datei[i] + "\r";
    }
    datei_txt.text = datei.name;
    upload_btn.enabled = true;
  } else {
    upload_btn.enabled = false;
    datei_txt.text = "";
    display_txt.text = "Wähle eine '*.as' Datei.";
  }
};
browseListener.onCancel = function () {
  upload_btn.enabled = false;
  display_txt.text = "Keine Datei ausgewählt -
  ➥ Bitte eine Datei wählen!";
}
browseListener.onHTTPError =
➥ function(fileRef, httpFehler) {
  display_txt.text = "Fehlernummer " + httpFehler
  ➥ + " während des Uploads";
}

// Multiple Auswahl von Dateien
fileRef = new flash.net.FileReferenceList ();

// Einfache Auswahl von Dateien
fileRef = new flash.net.FileReference ();
fileRef.addListener (browseListener);

onBrowse = function () {
  // Nutzer auffordern eine Datei auszuwählen
  fileRef.browse();
}
browse_btn.addEventListener ("click", onBrowse);

onUpload = function () {
  display_txt.text += "Upload initalisiert \n";
  display_txt.text += fileRef.upload + " \n";
  fileRef.upload("?");
}
upload_btn.enabled = false;
upload_btn.addEventListener ("click", onUpload);

// Daten der ausgewählten Datei
datei_daten =
➥ new flash.external.ExternalInterface ();
for (var i in datei_daten){
  display_txt.text += i + ": " + datei_daten[i]
  ➥ + "\r";
}
```

Schritt 3:

Damit die Dateien auf dem Server abgelegt werden können, wird noch ein serverseitiges Skript benötigt. Im vorliegenden Fall haben wir Ihnen hierfür PHP-Codezeilen zusammengestellt, die Sie in die Datei *index.php* übertragen müssen.

```
<!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>
<meta http-equiv="Content-Type" content="text/html;
➥ charset=iso-8859-1" />
<title>Upload via Flash - nur .as-Dateien</title>
</head>
<body bgcolor="#ffffff">
<?php

$uploadpfad = "uploads/";

foreach ($_FILES as $fieldName => $file) {
    echo move_uploaded_file($file['tmp_name'],
    ➥ $uploadpfad .$folder . $file['name']);
  }
?>
```

```
<object classid="clsid:d27cdb6e-ae6d-11cf-96b8-
➥ 444553540000" codebase="http://fpdownload.
➥ macromedia.com/pub/shockwave/cabs/flash/
➥ swflash.cab#version=8,0,0,0" width="550"
➥ height="400" id="uploadFiles" align="middle">
<param name="allowScriptAccess"
➥ value="sameDomain" />
<param name="movie" value="upload.swf" />
<param name="quality" value="high" />
<param name="bgcolor" value="#ffffff" />
<embed src="upload.swf" quality="high"
➥ bgcolor="#ffffff" width="550" height="400"
➥ name="uploadFiles" align="middle"
➥ allowScriptAccess="sameDomain" type=
➥ "application/x-shockwave-flash" pluginspage=
➥ "http://www.macromedia.com/go/getflashplayer" />
</object>
</body>
</html>
```

Schritt 4:

Sie sollten beim Testen darauf achten, dass Sie entweder über einen lokalen Testserver oder über einen Webspace samt PHP-Unterstützung bei einem Provider verfügen. Der Testdurchlauf selbst erfolgt innerhalb des Browsers. In der Adressleiste des Browsers muss die korrekte Adresse zum Testserver übergeben werden. Bei einem lokalen Testserver handelt es sich in den meisten Fällen um *http://localhost/verzeichnisname* oder *http://127.0.0.1/verzeichnisname*.

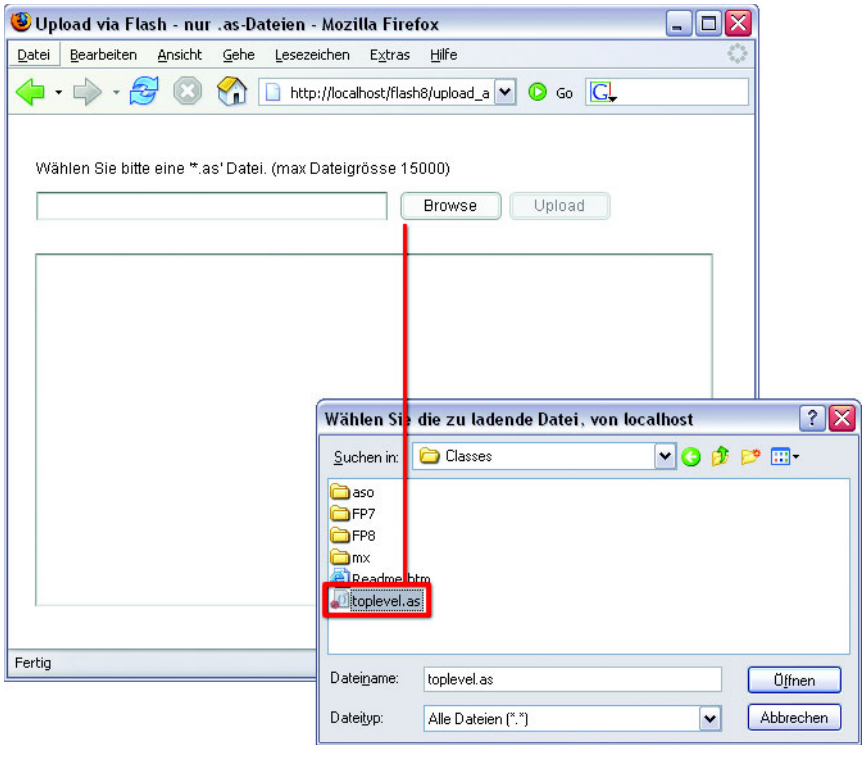


Abbildung 3.44: Klicken Sie auf den Browse-Schalter, um das Auswahl-Dialogfeld zu öffnen.

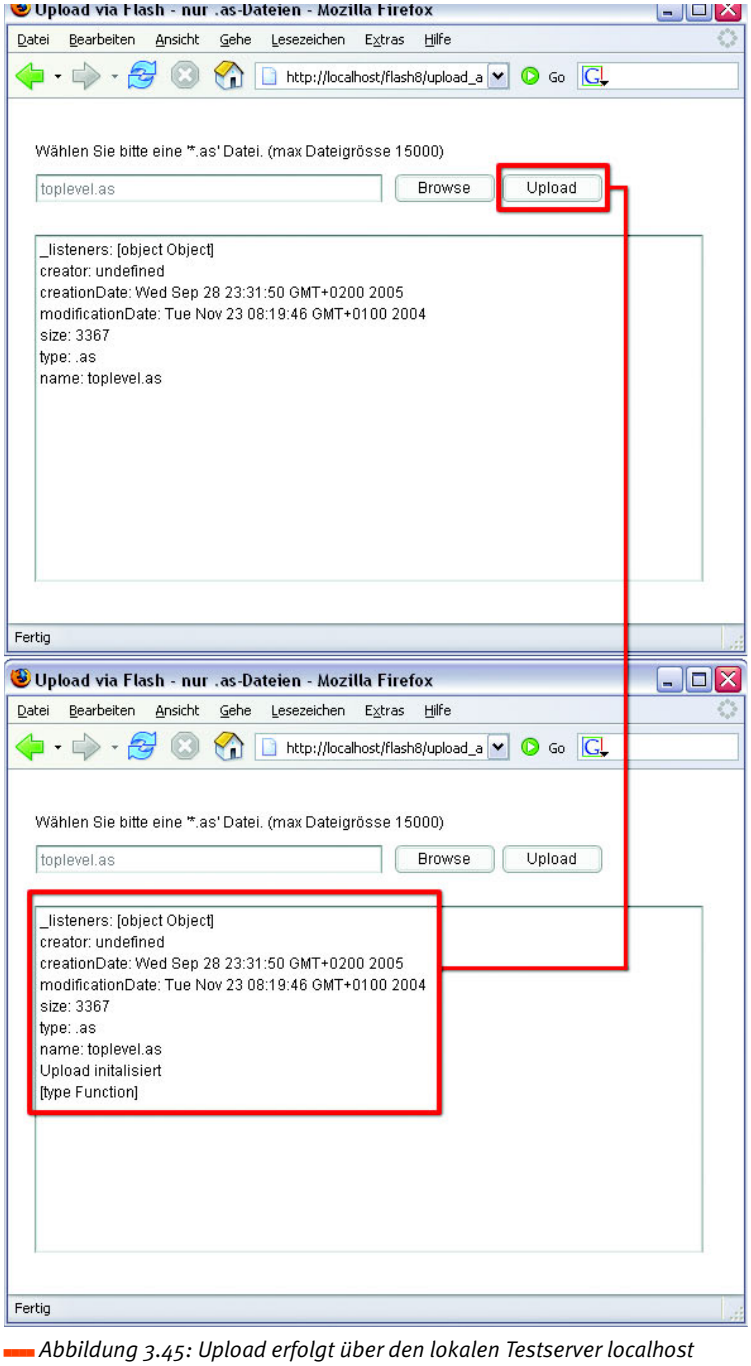


Abbildung 3.45: Upload erfolgt über den lokalen Testserver localhost

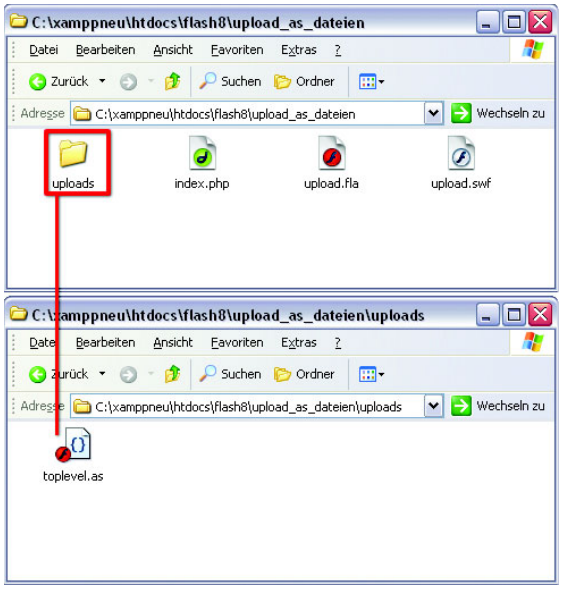


Abbildung 3.46: Die Dateien befinden sich nach dem Upload im Verzeichnis uploads.

3.1.11 Erweiterter Upload von Dateien via Flash & PHP

In diesem Workshop stellen wir Ihnen ein erweitertes Upload-Beispiel vor, das den direkten Abruf der hochgeladenen Dateien ermöglicht und in einem separaten Popup-Fenster zum Download anbietet. Auch in diesem Fall kommt Ihnen die `FileReference`-Klasse zu Hilfe.

▶ Schritt 1:

Wir benötigen als Erstes einen Flash-Film, diesen sollten Sie unter dem Namen *upload.fla* speichern. Aus dem Komponenten-Bedienfeld, das Sie mit Hilfe der Option Fenster/Komponenten erreichen, ziehen Sie zwei TextArea-Komponenten-Instanzen auf die Bühne und weisen Sie diesen die Instanznamen `dateiname` und `display_txt` zu. Zusätzlich benötigen Sie eine Button-Komponente. Weisen Sie dieser den Instanznamen `upload_btn` zu. Sie sollten auch noch zwei Movieclips *image* und *mask* zur Verfügung stellen, diese dienen als Bildbehälter und Maske.

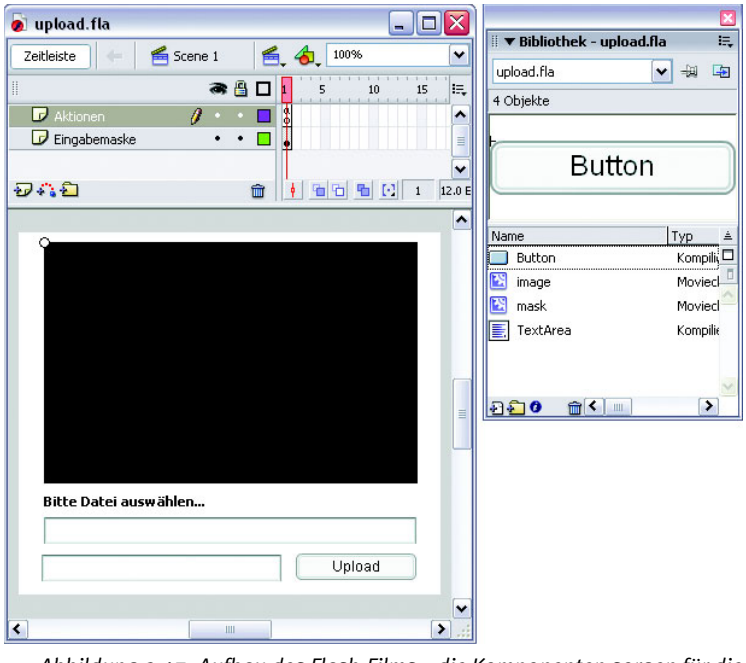


Abbildung 3.47: Aufbau des Flash-Films – die Komponenten sorgen für die Festlegung wichtiger Upload-Optionen

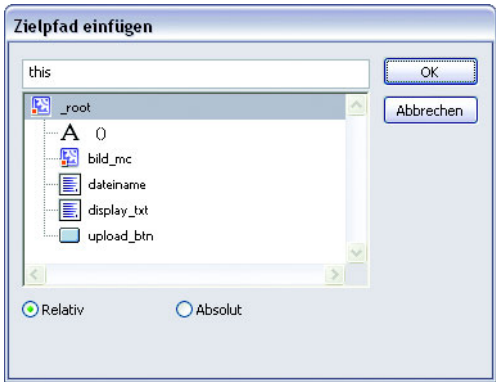


Abbildung 3.48: Sämtliche Instanznamen zur besseren Orientierung auf einen Blick

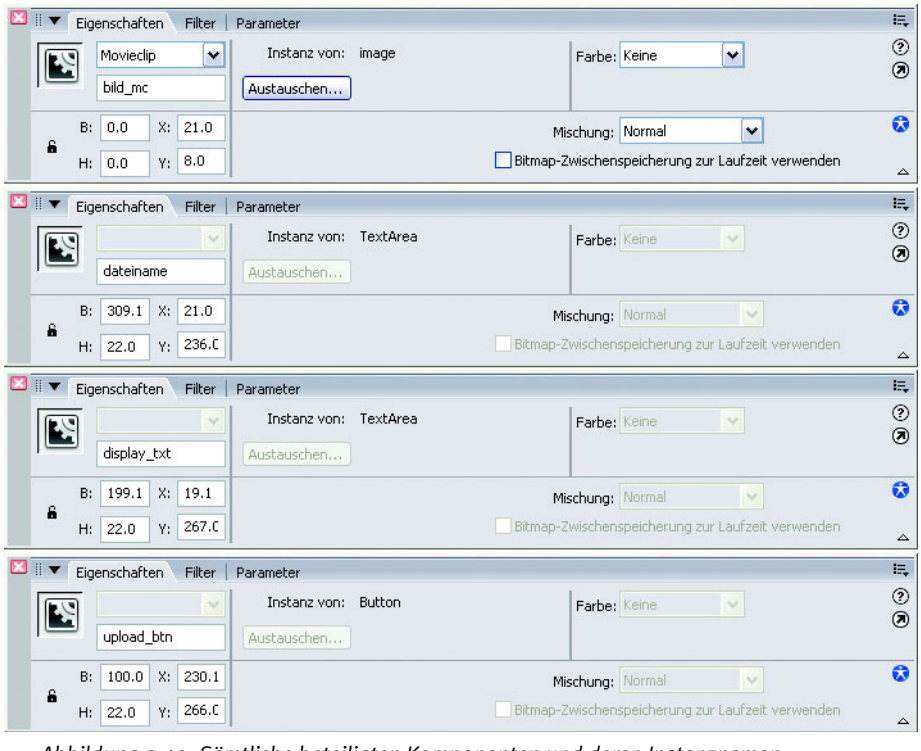


Abbildung 3.49: Sämtliche beteiligten Komponenten und deren Instanznamen

▶ Schritt 2:

Im ersten Schlüsselbild der Hauptzeitleiste platzieren Sie in der *Aktionen*-Ebene folgende ActionScript-Codezeilen.

```
import flash.net.FileReference;

var fileRef:FileReference = new FileReference();
var listener:Object = new Object();
var ziel = this;
var maxBreite = 310;
var maxHoehe = 200;
var pfad =
➥ "http://localhost/flash8/flashuploading/";

MovieClip.prototype.displayImage =
➥ function(url:String) {
  var target = this;
  var loader = new MovieClipLoader();
  loader.onLoadInit = function() {
    target._width = maxBreite;
    target._height = maxHoehe;
    ziel.attachMovie("mask","mask",
    ➥ this.getNextHighestDepth(),{_x: target._x,
    ➥ _y: target._y});
    target.setMask(ziel.mask);

    target.onRelease = function() {
      getURL(url,"_blank");
    }
  }
  loader.loadClip(url,target);
}

listener.onSelect = function(datei:FileReference) {
  upload_btn.enabled = false;
  dateiname.text = datei.name;
  datei.upload(pfad+"upload.php");
}

listener.onOpen = function(datei:FileReference) {
  display_txt.text = "Prüfe Daten...";
  // Dateigrössen-Limit (1 MB)
  var dateigroesse:Number =  int(datei.size/1024);
  if (dateigroesse>=1024) {
    display_txt.text = "Fehler: Grössenlimit 1 MB!";
    fileRef.removeListener(listener);
    _root.upload_btn.enabled = true;
  }
};

listener.onComplete =
➥ function(datei:FileReference) {
  display_txt.text = "Datei hochgeladen!";
  bild_mc.displayImage(pfad+"uploads/"+datei.name);

  bild_mc.onRelease = function() {
    getURL(pfad+"uploads/"+datei, "_blank");
  }
  _root.upload_btn.enabled = true;
}
fileRef.addListener(listener);

upload_btn.onRelease = function() {
  fileRef.addListener(listener);
  fileRef.browse([{description: "Bilddateien",
  ➥ extension: "*.jpg;*.gif;*.png", macType:
  ➥ "JPEG;jp2_;GIFF"}, {description: "Flash Filme",
  ➥ extension: "*.swf", macType: "SWFL"}]);
};
```

▶ Schritt 3:

Damit die Dateien auf dem Server abgelegt werden können, wird noch ein serverseitiges Skript benötigt. Im vorliegenden Fall haben wir Ihnen hierfür PHP-Codezeilen zusammengestellt, die Sie in die Datei *upload.php* übertragen müssen.

```
<?php
$uploadpfad = "/uploads/";
if($_FILES['Filedata']['size'] > 0 &&
➥ $_FILES['Filedata']['size'] < 1048576) {
  $pfad = dirname(__FILE__) . $uploadpfad .
  ➥ $_FILES['Filedata']['name'];
  move_uploaded_file($_FILES
  ➥ ['Filedata']['tmp_name'], $pfad);
}
?>
```

Schritt 4:

Sie sollten beim Testen darauf achten, dass Sie entweder über einen lokalen Testserver oder über einen Webspace samt PHP-Unterstützung bei einem Provider verfügen. Der Testdurchlauf selbst erfolgt innerhalb des Browsers. In der Adressleiste des Browsers muss die korrekte Adresse zum Testserver übergeben werden. Bei einem lokalen Testserver handelt es sich in den meisten Fällen um *http://localhost/verzeichnisname* oder *http://127.0.0.1/verzeichnisname*.

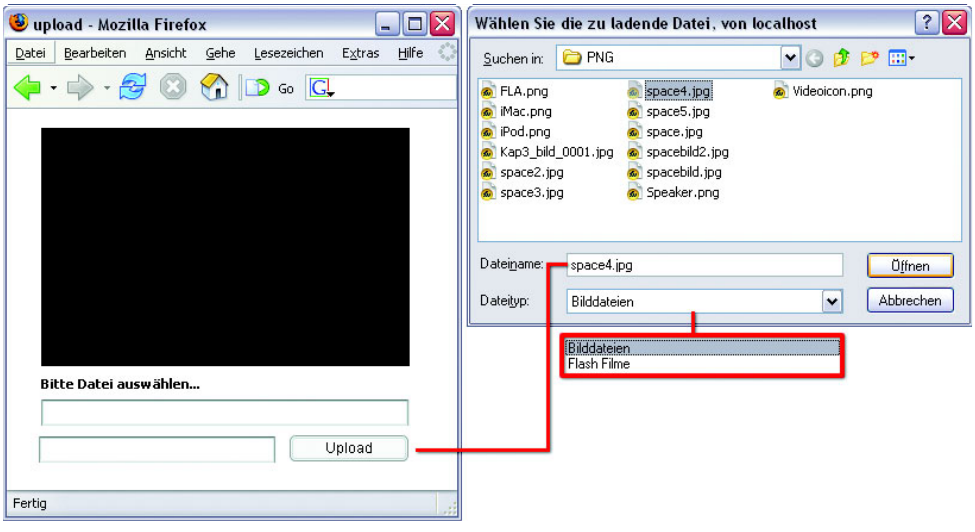


Abbildung 3.50: Upload erfolgt über den lokalen Testserver localhost.

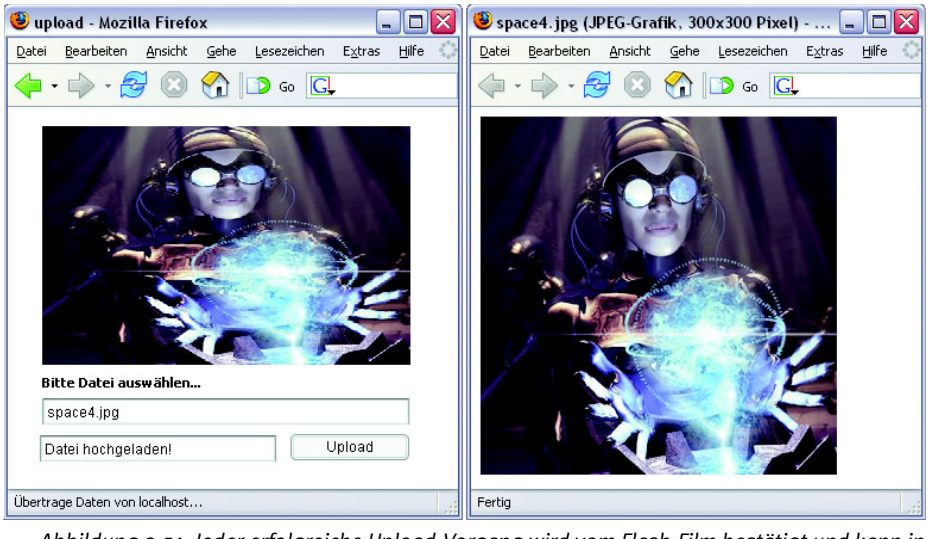


Abbildung 3.51: Jeder erfolgreiche Upload-Vorgang wird vom Flash-Film bestätigt und kann in einem separaten Popup-Fenster abgerufen werden.

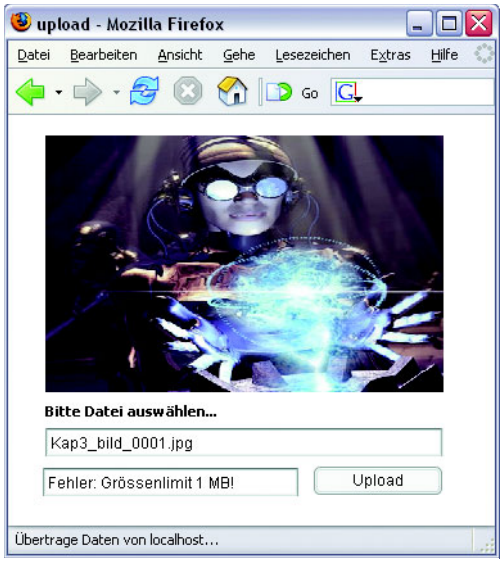


Abbildung 3.52: Dateien, die das Größenlimit überschreiten, müssen draußen bleiben.

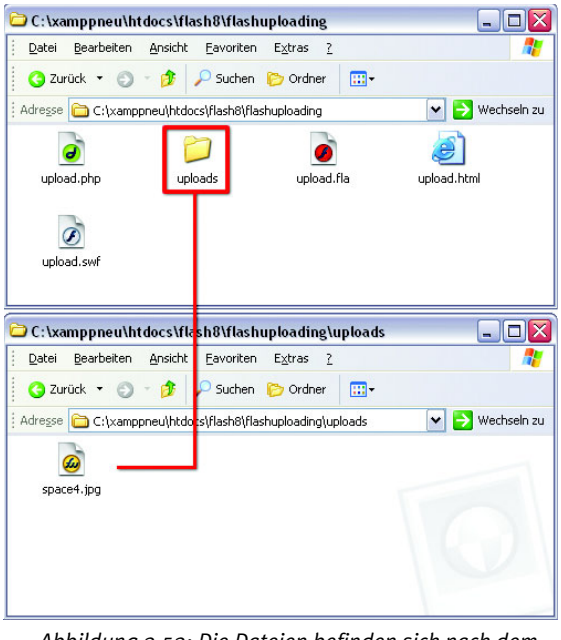


Abbildung 3.53: Die Dateien befinden sich nach dem Upload im Verzeichnis uploads.

3.1.12 Flash-Dashboard – erzeugen von Bilddateien via Flash & PHP

In diesem Workshop stellen wir Ihnen eine besondere Art von Zeichenbrett (Dashboard) vor. Das Dashboard versetzt Sie in die Lage, Ihre angefertigten Skizzen als Bilddateien auf dem Server zu speichern. Auch in diesem Fall verwenden wir eine PHP-basierte Lösung.

Schritt 1:

Wir benötigen als Erstes einen Flash-Film, diesen sollten Sie unter den Namen *index.fla* speichern. Aus dem Komponenten-Bedienfeld, das Sie mit Hilfe der Option Fenster/Komponenten erreichen, ziehen Sie zwei ComboBox-Komponenten-Instanzen auf die Bühne und weisen diesen die Instanznamen `format_cb` und `farben_cb` zu. Zusätzlich benötigen Sie eine Button-Komponente. Weisen Sie dieser den Instanznamen `erzeugen_btn` zu. Sie sollten auch noch die zwei Movieclips *Bildcontainer* und *Zielcontainer* zur Verfügung stellen, diesen weisen Sie die Instanznamen `bild_mc` und `ziel_mc` zu. Eine TextArea-Komponente mit dem Instanznamen `status_txt` wird noch benötigt, um die Ausgabe der diversen Meldungen sicherzustellen. Die NumericStepper-Komponente erhält den Instanznamen `pixelstepper`, hierüber lässt sich die Pixelstärke beeinflussen.

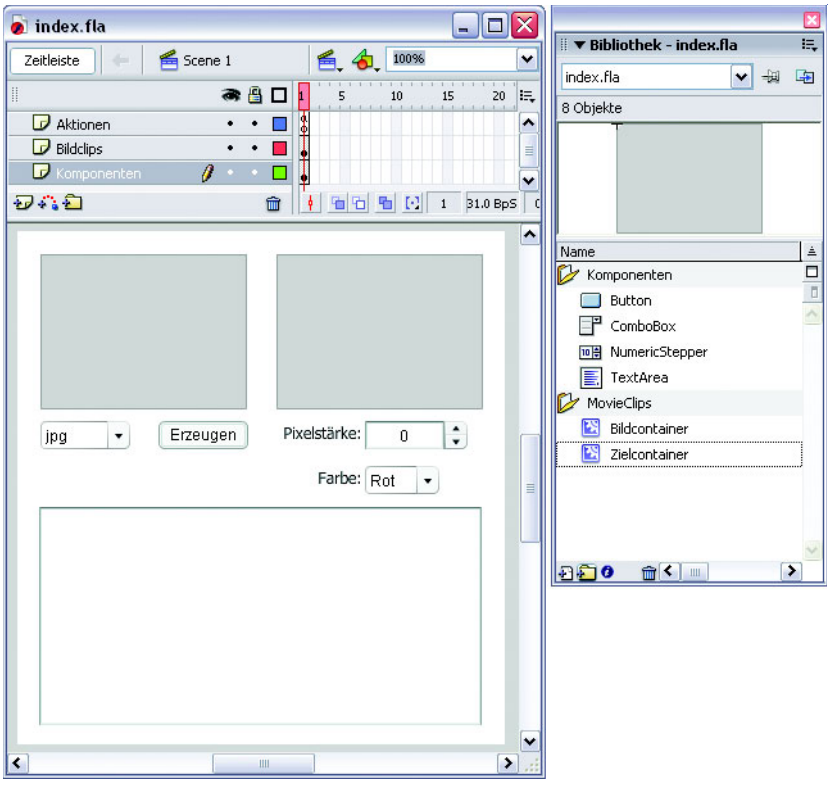


Abbildung 3.54: Aufbau des Flash-Films – die Komponenten liefern die Eingabemaske

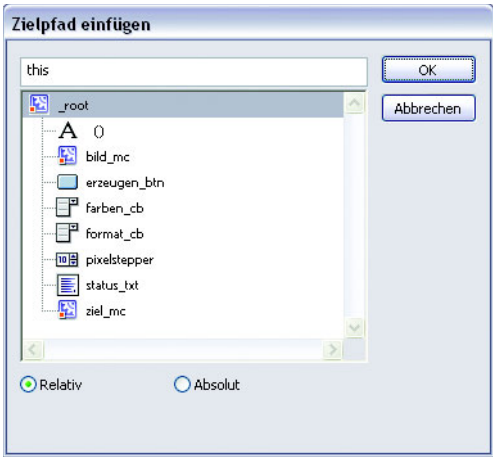


Abbildung 3.55: Sämtliche Instanznamen zur besseren Orientierung auf einen Blick

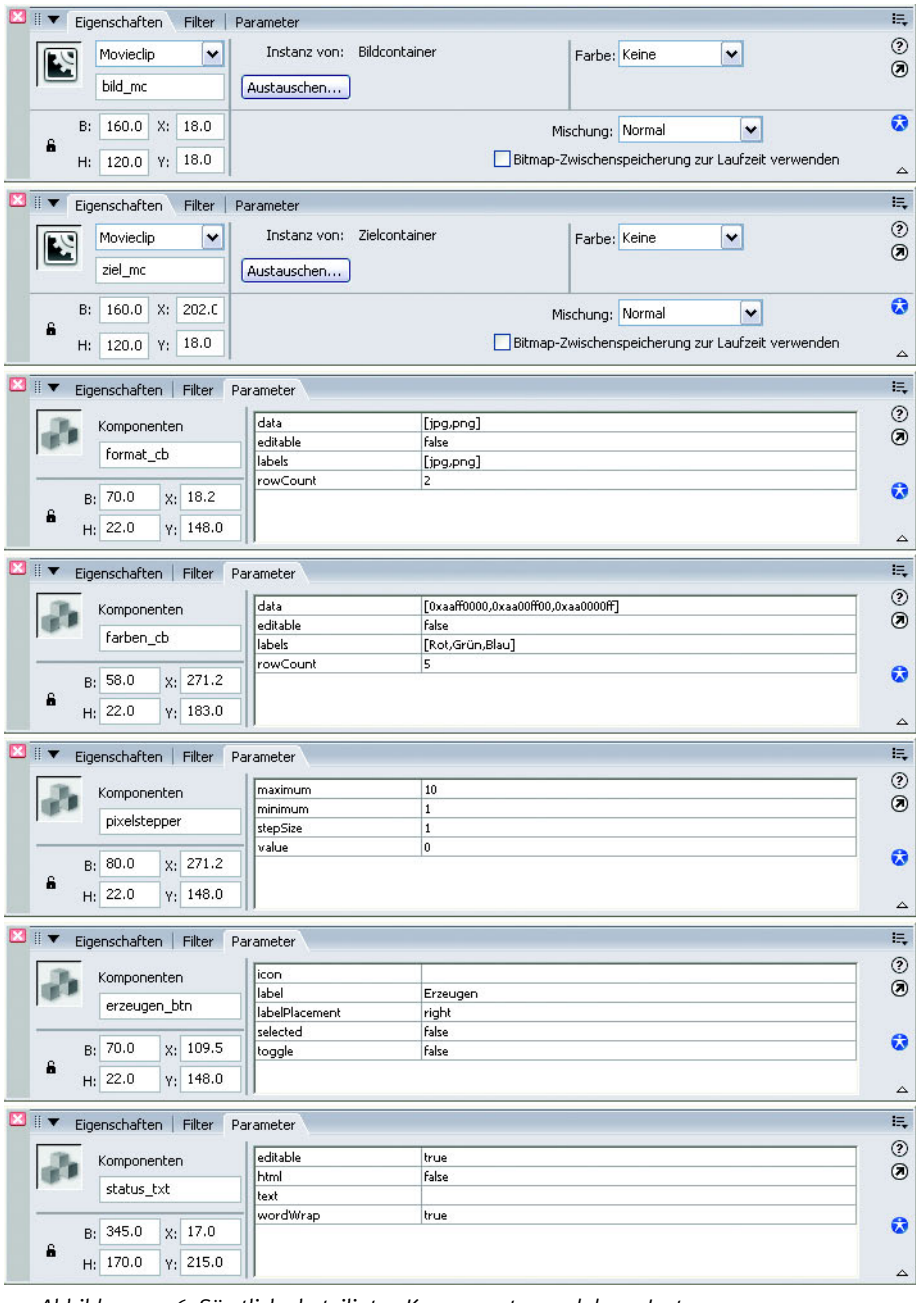


Abbildung 3.56: Sämtliche beteiligten Komponenten und deren Instanznamen

▶ **Schritt 2:**

Im ersten Schlüsselbild der Hauptzeitleiste platzieren Sie in der *Aktionen*-Ebene folgende ActionScript-Codezeilen.

```
var breite:Number = bild_mc._width;
var hoehe:Number = bild_mc._height;

bild_mc.cacheAsBitmap = true;

var bilddaten = new flash.display.BitmapData
➥ (breite, hoehe, true, 0x10ffffff);
bilddaten.transparent = true;
bilddaten.draw(bild_mc);

bild_mc.swapDepths(100);

erzeugen_btn.onRelease = function() {
  var startzeit:Date = new Date();
  var bildzeichenkette:String = "";

  for (var i = 0; i<breite; i++) {
    for (var j = 0; j<hoehe; j++) {
      var _local1 = bilddaten.getPixel(i, j);
      tr = _local1 >> 16;
      tg = (_local1 >> 8) ^ (tr << 8);
      tb = _local1 ^ ((tr << 16) | (tg << 8));
      r = tr.toString(16);
      g = tg.toString(16);
      b = tb.toString(16);
      while (r.length<2) {
        r = "0"+r;
      }
      while (g.length<2) {
        g = "0"+g;
      }
      while (b.length<2) {
        b = "0"+b;
      }
      bildzeichenkette += r+""+g+""+b+" ";
    }
  }
  this._parent.zeitstempel = new Date().getTime();

  datenobjekt = new LoadVars();
  datenobjekt.breite = breite;
  datenobjekt.hoehe = hoehe;
  datenobjekt.uploaddaten = bildzeichenkette;
  datenobjekt.zeitstempel = zeitstempel;
  datenobjekt.dateiendung =
  ➥ format_cb.selectedItem.data;
  var endzeit:Date = new Date();
  var zeit:Number = (endzeit-startzeit)/1000;
  status_txt.text = "Berechnet in "+zeit+
  ➥ " Sekunden";
  status_txt.text += "\nDaten werden an den
  ➥ Server übertragen...";
  datenobjekt.sendAndLoad("erzeugen.php?zufall="
  ➥ +this._parent.zeitstempel, datenobjekt,
  ➥ "POST");
  datenobjekt.onLoad = function(success) {
    if (success) {
      var endzeit:Date = new Date();
      var zeit:Number = (endzeit-startzeit)/1000;
      status_txt.text += "\nÜbertragung beendet /
      ➥ GD Berechnung in "+zeit+" Sekunden";
      var loader:MovieClipLoader =
      ➥ new MovieClipLoader();
      loader.addListener(ladeStatus);
      ziel_mc.duplicateMovieClip("ladebalken", 10,
      ➥ {_x:_root.ziel_mc._x,
      ➥ _y:_root.ziel_mc._y});
      datei = "uploads/bild"+zeitstempel+"."
      ➥ +format_cb.selectedItem.data;
      status_txt.text += "\nLade "+datei+"...";
      loader.loadClip(datei, ziel_mc);
      _root.datei = datei;
    } else {
      status_txt.text += "Fehler!";
    }
  };
};
```

```
var ladeStatus:Object = new Object();
ladeStatus.onLoadProgress = function(ziel_mc:
➥ MovieClip, bytesLoaded:Number,
➥ bytesTotal:Number) {
  var prozent:Number =
  ➥ int(bytesLoaded/bytesTotal*100);
  ladebalken._xscale = prozent;
};

ladeStatus.onLoadComplete = function(ziel_mc:
➥ MovieClip) {
  _root.ladebalken.removeMovieClip();
  status_txt.text += "Erledigt!";
  status_txt.text += "\nBitte auf das Bild klicken,
  ➥ um es herunterzuladen "
  ➥ +format_cb.selectedItem.data+"!";
  ziel_mc.onRelease = function() {
    fileRef = new flash.net.FileReference();
    dateiname = _root.datei;
    fileRef.addListener({onComplete:function (d) {
      onComplete(d);
    }, onProgress:function (d, l, t) {
      onProgress(d, l, t);
    }, onSelect:function (d) {
      onSelect(d);
    }});
    fileRef.download(dateiname);
    function onComplete() {
      status_txt.text += "\nBilddownload beendet!";
    }
    function onProgress(d, bytesLoaded:Number,
    ➥ bytesTotal:Number) {
      status_txt.text += ".";
    }
    function onSelect(d) {
      status_txt.text += "\nDownload...";
    }
  };
};

bild_mc.onRollOver = function() {
  _root.onMouseMove = function() {
    if (zeichnen) {
    bild_mc.setMask(null);
    for (var i = 0; i<pixelstepper.value; i++) {
      for (var j = 0; j<pixelstepper.value; j++) {
        bilddaten.setPixel32(bild_mc._xmouse+i,
        ➥ bild_mc._ymouse+j, farben_cb.value);
      }
    }
    bild_mc.attachBitmap(bilddaten, 10, "auto",
    ➥ true);
    }
  };

};
bild_mc.onRollOut = function() {
  bild_mc.onMouseMove = undefined;
};

bild_mc.onPress = function() {
  zeichnen = true;
}

bild_mc.onRelease = bild_mc.onReleaseOutside =
➥ function() {
  zeichnen = false;
}
```

Schritt 3:

Damit die Bilder auf dem Server erzeugt werden können, wird noch ein serverseitiges Skript benötigt. Im vorliegenden Fall haben wir Ihnen hierfür PHP-Codezeilen zusammengestellt, die Sie in die Datei *erzeugen.php* übertragen müssen. Bitte beachten Sie, dass zur dynamischen Erzeugung von Bilddateien die GD-PHP-Erweiterung auf dem Testserver vorhanden sein muss.

```
<?
function zeichneBild() {
  $pbreite = intval($_POST['breite']);
  $phoehe = intval($_POST['hoehe']);
  $puploaddaten = $_POST['uploaddaten'];
  $pendung = $_POST['dateiendung'];
  $pzeitstempel = $_POST['zeitstempel'];
  $bild = imagecreatetruecolor($pbreite,$phoehe);
  $farbe = explode(" ", $puploaddaten);
  $i=0;
  $j=0;
  $n=0;
  while ($i<$pbreite) {
    while ($j<$phoehe) {
      // Konvertierung HEX in DECIMAL
      $hexfarbe = str_split($farbe[$n], 2);
      $r = hexdec($hexfarbe[0]);
      $g = hexdec($hexfarbe[1]);
      $b = hexdec($hexfarbe[2]);
      // Farbe ins Bild hinzufügen
      $tmp = ImageColorAllocate($bild,$r, $g, $b);
      // Pixel festlegen
      imagesetpixel($bild,$i, $j, $tmp);
      $j++;
      $n++;
    }
    $i++;
    $j = 0;
  }
  // JPEG
  if ($pendung == "jpg") {
  imagejpeg($bild,
  ➥ 'uploads/bild'.$pzeitstempel.'.jpg', 100);
  }
  // PNG
  if ($pendung == "png") {
  imagepng($bild,
  ➥ 'uploads/bild'.$pzeitstempel.'.png', 100);
  }
  imagedestroy($bild);
  echo "true";
}
zeichneBild();
?>
```

Schritt 4:

Sie sollten beim Testen darauf achten, dass Sie entweder über einen lokalen Testserver oder über einen Webspace samt PHP-Unterstützung bei einem Provider verfügen. Der Testdurchlauf selbst erfolgt innerhalb des Browsers. In der Adressleiste des Browsers muss die korrekte Adresse zum Testserver übergeben werden. Bei einem lokalen Testserver handelt es sich in den meisten Fällen um *http://localhost/verzeichnisname* oder *http://127.0.0.1/verzeichnisname*.

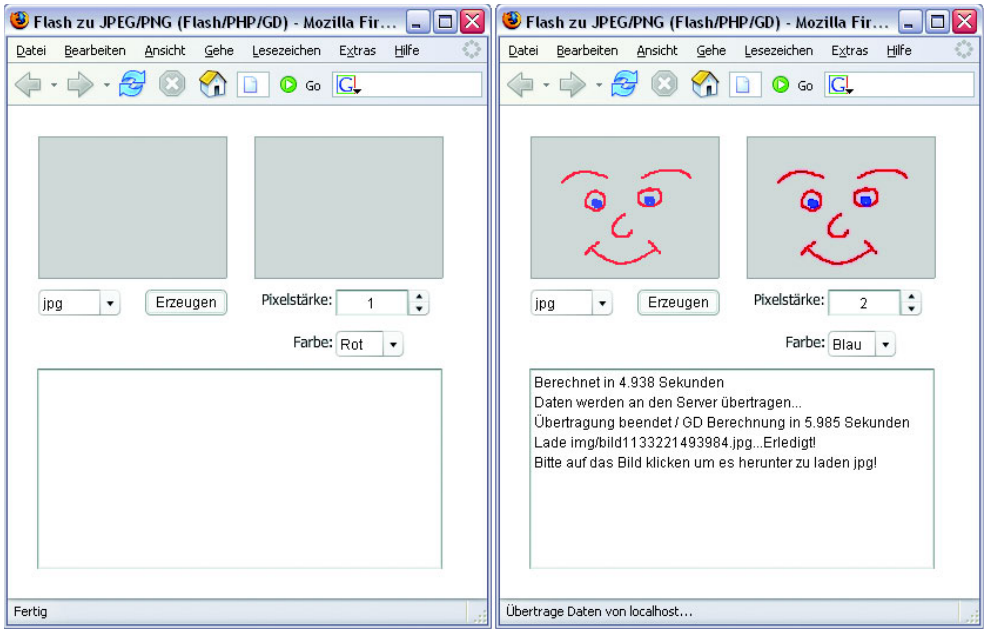


Abbildung 3.57: Lassen Sie Ihrer Fantasie freien Lauf – nur zeichnen und einmal auf die Schaltfläche Erzeugen *klicken.*

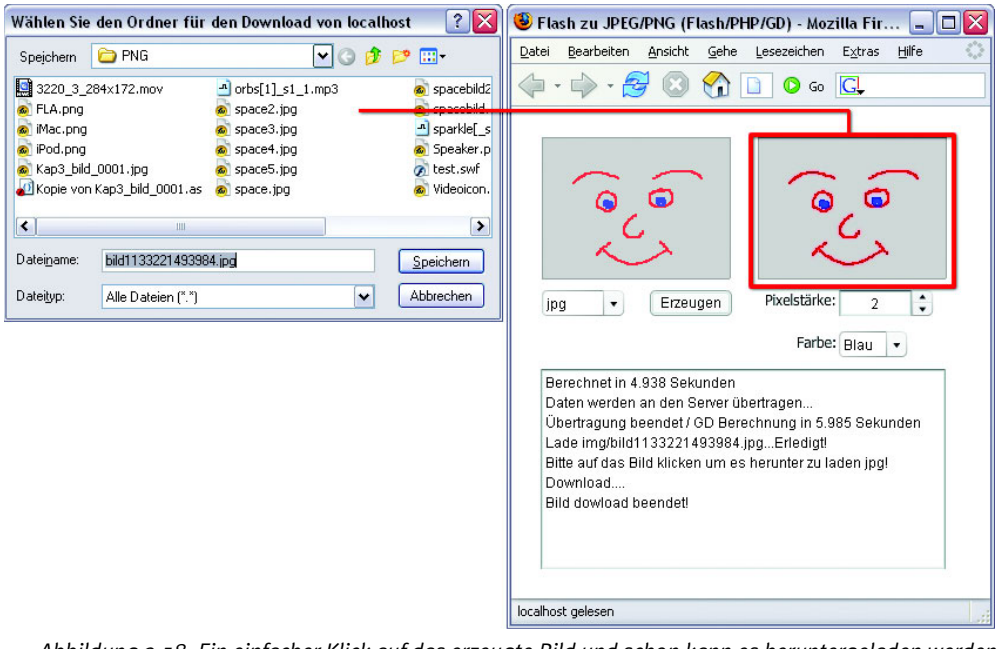


Abbildung 3.58: Ein einfacher Klick auf das erzeugte Bild und schon kann es heruntergeladen werden.

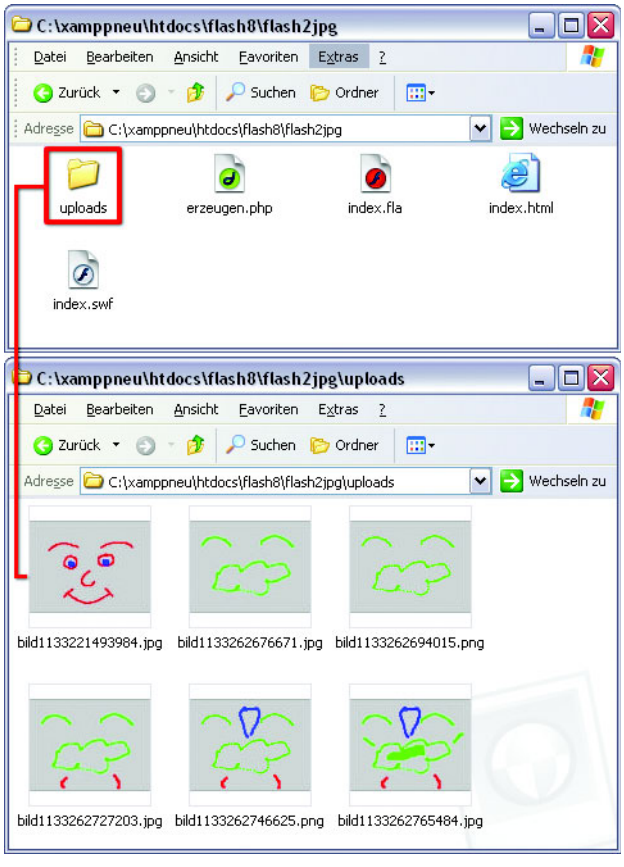


Abbildung 3.59: Die dynamisch erzeugten Bilddateien befinden sich im uploads-Verzeichnis.