

edition **PAGE**



Kluth, Kyeck,
Schricker, Woost

Flash fast forward

Die Workshops für Fortgeschrittene

Aktuell
zu Adobe
Flash CS3



- Alle Beispieldateien zum Nachbauen
- 30-Tage-Testversion Flash CS3 (Windows und Mac)
- Über 1 Stunde Video-Lektionen zu ActionScript 3

Galileo Design

Sönke Kluth, Frederik Schricker,
Philipp Kyeck, Aron Woost

Flash fast forward

Die Workshops für Fortgeschrittene

Liebe Leserin, lieber Leser,

sicher haben auch Sie schon oft im Internet einen tollen Flash-Effekt entdeckt und sich dann auf die Suche nach einem Tutorial gemacht, um ihn nachzubauen. Nur leider haben Sie im Netz keine Anleitung gefunden, die Sie zum Ziel führt und auch Ihre Fragen in den Foren wurden nur unzureichend beantwortet ...

Dann werden Sie staunen, wenn Sie sich die Workshops in diesem Buch anschauen: Die vier Autoren haben die schönsten Effekte, raffiniertesten Anwendungen und effizientesten Lösungen für Sie zusammengestellt. Schritt-für-Schritt kommen Sie hier wirklich ans Ziel. In insgesamt sechs Teilen finden Sie die unterschiedlichsten Workshops: Mal erzeugen Sie Schneefall, hängende Kabel und physikalisch korrekt hüpfende Bälle. Mal nutzen Sie die Bitmap-Klasse für Partikeleffekte und Flüssigkeitssimulationen. Oder Sie programmieren eigene Klassen für komplexe Flash-Anwendungen wie ein XML-Menü oder eine Slideshow und erzeugen 3D-Räume. Auch PHP und MySQL kommen zum Einsatz, Audio und Video werden dynamisch eingebunden und schließlich lernen Sie, wie Sie Ihre Flash-Anwendungen professionell im Internet veröffentlichen.

Alle Beispiele finden Sie auf der Buch-DVD als fla- und als swf-Datei. So können Sie bereits am Anfang eines Workshops sehen, wie das Beispiel funktioniert und den ActionScript-Code einfach kopieren. Bei komplexeren Workshops haben die Autoren zusätzlich Zwischenschritte abgespeichert. Als besonderes Special finden alle, die sich richtig in die neue Programmiersprache von Flash einarbeiten wollen, auf der Buch-DVD ausgewählte Video-Lektionen zu ActionScript 3.

Nun bleibt mir noch, Ihnen viel Freude beim Nachbauen der Workshops zu wünschen. Lassen Sie sich inspirieren!

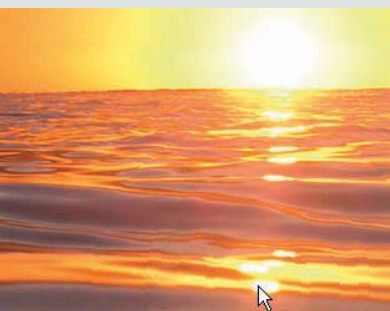
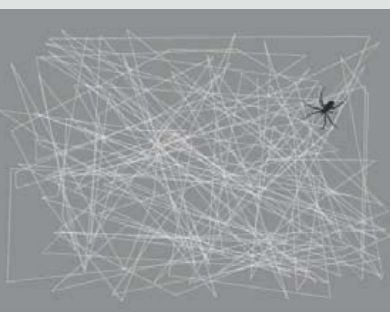
Katharina Geißler

Lektorat Galileo Design

katharina.geissler@galileo-press.de

Galileo Press • Rheinwerkallee 4 • 53227 Bonn

www.galileodesign.de



Inhalt

Vorwort	8

Simulationen & Effekte

Klassen, Vererbung, Packages	12
<i>Exkurs</i>	
Schrift-Effekt	17
<i>Schrift wie von Hand zeichnen lassen</i>	
Schreibmaschinen-Effekt	24
<i>Einen klassischen Effekt in einer Klasse realisieren</i>	
Zufallsanimation	29
<i>Eine Spinne krabbelt zufallsgesteuert hin und her</i>	
Scroll-Effekt	42
<i>Flash-Filter für eine weichgezeichnete Scrollbewegung</i>	
Wellen simulieren	51
<i>Kreativer Einsatz von Masken</i>	
Pflanzengenerator	58
<i>Pflanzen mit der Maus wachsen lassen</i>	

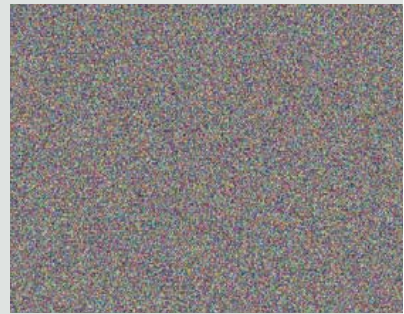
Effekte mit der Bitmap-Klasse

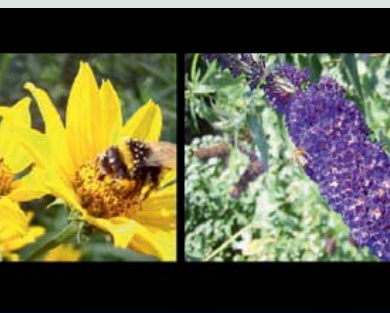
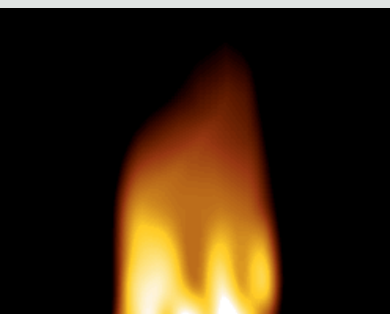
Das BitmapData-Objekt	72
<i>Exkurs</i>	
TV-Rauschen	75
<i>Das Rauschen eines Fernsehers simulieren</i>	
Verwischen-Effekt	79
<i>Bitmapdaten kopieren und verwischen</i>	
Vergrößerungs-Effekt	87
<i>Bildteile wie mit einer echten Lupe vergrößern</i>	

Farbwähler	96
<i>Farbwerte eines Bildes auslesen und anzeigen</i>	
Anzeigeliste und attachMovie()	104
<i>Exkurs</i>	
Partikel-Effekt	107
<i>Bildpartikel animieren</i>	
Flüssigkeit simulieren	120
<i>Wasserspritzer mit der Maus erzeugen</i>	
Feuer-Effekt	131
<i>Ein feuriges Flash-Vergnügen mit den Bitmap-Operationen</i>	
Web 2.0-Reflexion	141
<i>Movieclips spiegeln</i>	

Flash-Physics und 3D-Programmierung

Schneefall simulieren	152
<i>Fallenden Schnee im Wind simulieren</i>	
Springende Bälle	163
<i>Physikalische Eigenschaften programmieren</i>	
Kabelschwingung	175
<i>Ein bewegliches Kabel programmieren</i>	
3D-Programmierung in Flash CS3	184
<i>Exkurs</i>	
3D-Sternenfeuer	188
<i>Eine 3D-Engine programmieren</i>	
3D-Bildergalerie	198
<i>Im Raum fliegende Bilder</i>	
3D-Karussellmenü	218
<i>Menüelemente im dreidimensionalen Raum</i>	





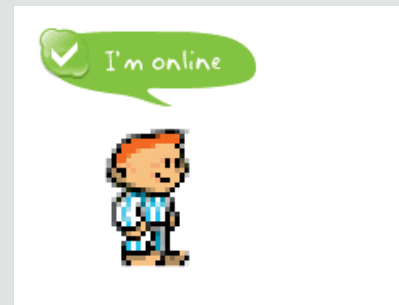
Anwendungen und dynamische Inhalte

ClipLoader	234
<i>Externe Bilder über eine eigene Loader-Klasse laden</i>	
Doppelklick-Klasse	245
<i>Bilder per Doppelklick ein- und auszoomen</i>	
Tooltips	253
<i>Programmieren einer eigenen Tooltip-Klasse</i>	
Dateiupload	263
<i>Dateien auf einen Server übertragen</i>	
Individuelle Kontextmenüs	268
<i>Programmieren einer eigenen Kontextmenü-Klasse</i>	
Laufband	279
<i>Ein Bilder-Laufband mit der Maus steuern</i>	
Slideshow	290
<i>Bilder in einer Diashow präsentieren</i>	
XML	300
<i>Exkurs</i>	
XML-Menü	302
<i>Ein Menü extern über XML steuern</i>	
Flash Remoting	309
<i>Exkurs</i>	
Flash-Gästebuch	311
<i>Flash Remoting für dynamische Inhalte</i>	

Audio und Video

Audio-Player	342
<i>Ein eigener MP3-Player in Flash CS3</i>	
Soundspektrum	370
<i>Audio-Player um eine Spektrumsanzeige erweitern</i>	

Adobe AIR	377
<i>Exkurs</i>	
AIR-Musikplayer	380
<i>Aus dem MP3-Player wird ein vollwertiges Offline-Programm</i>	
Video-Player	387
<i>Flash-Videos im eigenen Player abspielen</i>	
 Flash im Internet	
Flash-Filme in HTML einbinden	406
<i>Mit dem SWFObject unnötige Klicks vermeiden</i>	
Zentriert, transparent und in High-Quality	411
<i>Veröffentlichungsparameter beeinflussen</i>	
Flash Player-Versionserkennung	418
<i>Automatische Erkennung und Installation des Flash Players</i>	
Kommunikation mit HTML	425
<i>Variablen zwischen HTML und Flash übergeben</i>	
Eine Website komplett in Flash	433
<i>Deeplinks, Browsernavigation und benutzerfreundliche URLs</i>	
Skype-Status	453
<i>Den aktuellen Skype-Status auf Ihren Webseiten integrieren</i>	
Die DVD zum Buch	464
Index	466



Vorwort

Als Flash-Designer und -Entwickler surft man ein wenig anders durch das Internet als der »normale« Websitebesucher. Denn man achtet nicht nur auf Inhalte und Designs, sondern immer auch auf schöne Effekte und Lösungen, die man in der Zukunft vielleicht einmal selbst in einem Projekt einsetzen könnte. So wird jeder Ausflug ins Internet zu einer Inspirationsreise in Sachen Flash.

In diesem Buch bieten wir Ihnen eine Auswahl von Anwendungen, Effekten und Lösungen, die Sie vielleicht schon einmal gesehen haben oder vielleicht – und das wäre natürlich noch besser! – auch nicht. Dabei haben wir versucht, eine spannende Mischung aus visuellen Effekten und technischen Lösungen zusammenzustellen. Somit ist sowohl etwas für den versierten Motion-Designer als auch für den klassischen Flash-Entwickler dabei.

Jeder Workshop kann eigenständig bearbeitet werden. Sie müssen das Buch also nicht »von vorn nach hinten« lesen, um die Workshops nachvollziehen zu können. Suchen Sie sich nach Lust und Laune einen Workshop heraus; arbeiten Sie ihn durch; und lernen Sie! Sie sollen sich mit Spaß und schnellen Erfolgserlebnissen neues Wissen aneignen und erarbeiten!

Das größte Kompliment für uns wäre, wenn ein »Grafik-Flasher« eine Backend-Verbindung mit dem Server herstellt und ein »Flash-Nerd« seinen Chef mit beeindruckenden grafischen Effekten überrascht.

Ist dieses Buch das richtige für mich?

Wenn Sie Flasher sind lautet die Antwort eindeutig: ja! Inzwischen hat sich das Anwendungsgebiet von Flash in ganz unterschiedliche Richtungen entwickelt. Da gibt es die Flash-Programmierer, die Online-Games und komplexe, an Content-Management-Systeme angebundene Websites entwickeln. Ihnen gegenüber stehen Flash-Designer, die Entwürfe in aufregende, lebendige Flash-Auftritte umsetzen. Und dann gibt es die vielen Spezialisten, die sich ein ganz bestimmtes Gebiet innerhalb der Flash-Welt gesucht haben, in dem sie »top« sind. Das sind Cartoon-Animatoren, Flex-Entwickler, AIR-Producer, Sounddesigner u. v. m.

Alle diese Gruppen werden ihren Spaß an diesem Buch haben. Und das nicht unbedingt, weil aus ihrem Spezialgebiet ein oder zwei Workshops dabei sind. Sondern weil sie – mit ihrem Flash-Wissen – in diesem Buch eine Menge über die anderen Anwendungsgebiete von Flash erfahren und das Gelernte in ihrer täglichen Praxis einsetzen können.

Was ist noch wichtig?

Wir Autoren können Kompromisse bei Form und Inhalt nicht immer vermeiden. Es ist für die Leser einfach zu wenig anschaulich, wenn sich die Workshops so lesen, wie man ein Projekt »in echt« umsetzen würde. Darum wurden alle Workshops speziell auf Lesbarkeit und Verständlichkeit hin geprüft und angepasst. Wenn Sie sich als absoluter Flash-Profi an der einen oder anderen Stelle fragen, ob man dies oder das nicht auch noch effizienter umsetzen könnte: ja, man könnte! :-). Aber uns war es wichtig, keine Leser auszuschließen – deshalb sind die Workshops alle so verständlich wie möglich geschrieben.

Mögliche Sicherheitswarnung

Bei einigen Workshops greifen Sie auf Skripte zu, die sich nicht bei Ihnen lokal auf dem Computer befinden. Dabei kann es passieren, dass die Sicherheitswarnung wie in Abbildung 1 erscheint.

Jetzt können Sie entweder diesen Zugriff mit Klick auf OK einmalig bestätigen oder Sie passen die Einstellungen an. Bei letzterem werden Sie auf eine Seite von Adobe geführt, die wie in Abbildung 2 aussieht.

Hier können Sie im dritten Reiter die globalen Sicherheitseinstellungen anpassen. Dafür wählen Sie zuerst Immer zulassen aus und suchen über das Dropdown-Menü Bearbeiten den Ordner aus, in dem die Beispieldateien liegen. So haben Sie die Zugriffsrechte neu gesetzt und werden die Sicherheitswarnung in Zukunft nicht mehr sehen.

Danke

Unser Dank geht an forst für Gestaltung aus Hamburg, die das schicke Layout für den Audio-Player gestaltet haben, Felix Jork (Fotodesign-Jork), der einige seiner Fotos bereitgestellt hat, und Babette Rösch, die einige der Grafiken im Buch erstellt hat. Weiterhin möchten wir allen Leuten danken, die dieses Buch ermöglicht haben. Dazu zählen Katharina, Gregor, Chi-Dong, Catrin, Julia, Hanno, Nikl, Nils, Andreas, Al, Sanjay und Jan.



▲ Abbildung 1
Sicherheitswarnung



▲ Abbildung 2
Sicherheitseinstellungen für den Flash Player

Simulationen & Effekte

Klassen, Vererbung, Packages	12
Schrift-Effekt.....	17
Schreibmaschinen-Effekt.....	24
Zufallsanimation	29
Scroll-Effekt	42
Wellen simulieren	51
Pflanzengenerator	58

Klassen, Vererbung, Packages

Video-Training

Auf der Buch-DVD finden Sie eine Video-Lektion zu den Grundlagen der objektorientierten Programmierung (Lektion 3.1).

Falls Sie sich bereits mit allen drei Versionen von ActionScript befasst haben, sind Ihnen sicherlich die teilweise gravierenden Unterschiede aufgefallen. Neben vielen anderen Änderungen hat sich ActionScript im Laufe der Zeit zu einer objektorientierten Sprache entwickelt, die auf Klassen und Objekten basiert. Damit einher gehen Vererbung und Packages.

Vielleicht kennen Sie die `prototype`-Anweisung aus ActionScript 1. Mit ihr war es möglich, eine bestehende Klasse, wie zum Beispiel `MovieClip`, zu erweitern. Das bedeutet: eine neue Art von Movieclips mit angepassten Funktionalitäten und Eigenschaften zu definieren, um sie fortan in einem Projekt verwenden zu können. Hierbei wurde ein Grundprinzip der objektorientierten Programmierung simuliert, die Vererbung. Dennoch hatte ActionScript 1 keine formell objektorientierte Syntax. Echtes objektorientiertes Programmieren wurde mit ActionScript 2 möglich, aber nicht zwingend erforderlich. Mit ActionScript 3 schließlich fiel die Abwärtskompatibilität dem Rotstift zum Opfer. Das Ergebnis ist eine rein objektorientierte Sprache.

Doch fangen wir von vorn an: Was bedeutet »objektorientiert« eigentlich?

Objektorientierte Programmierung

Wie der Name bereits vermuten lässt, basiert die objektorientierte Programmierung auf Objekten. Wenn Sie ein Programm schreiben, werden Sie irgendeine Funktionalität vor Augen haben, die Sie mit Ihrem Programm umsetzen wollen. Der Ansatz der objektorientierten Programmierung ist nun, diese Funktionalität gleich in der Konzeptionsphase in kleinere Teilfunktionalitäten zu zerlegen und aus diesen dann die Gesamtfunktionalität aufzubauen. Eine solche Teilfunktionalität entspricht dann, grob gesagt, einem Objekt. Der Vorteil liegt auf der Hand: Wenn Sie bereits eine Reihe von objektorientierten Programmen geschrieben haben, werden Sie früher oder später auf bereits bekannte Anforderungen stoßen. Diese müssen Sie jetzt nicht erneut programmieren. Sie können die Elemente Ihrer bisherigen Programme wiederverwenden.

Bevor sich der objektorientierte Ansatz durchsetzte, gab es die sogenannte »prozedurale Programmierung«. Hierbei wurde sämtlicher Code sozusagen einfach untereinander weg geschrieben. Sie können sich vielleicht vorstel-

len, wie das dann bei größeren Anwendungen aussah oder welchen Zeitaufwand es bedeutete, den Fehler zu suchen, wenn etwas nicht funktionierte. Bei der objektorientierten Programmierung hingegen wird der Code in Klassen aufgeteilt, von denen dann Objekte erstellt werden können – womit wir beim nächsten Punkt wären.

Klassen und Objekte

Klassen und Objekte sind untrennbar miteinander verbunden. Genauer gesagt gibt es zu jedem Objekt eine Klasse, die sozusagen den Bauplan des Objektes darstellt. Von dieser Klasse können dann beliebig viele Objekte – auch »Instanzen« genannt – erstellt und eingesetzt werden. Erst durch diese Instantiierung passiert irgendetwas. Der Code einer Klasse wird erst ausgeführt, wenn mindestens ein Objekt der Klasse erzeugt wurde.

Stellen Sie sich vor, Sie wollen eine Anwendung schreiben, in der Autos eine Rolle spielen. Da Autos immer wieder vorkommen, ist es sinnvoll, ein Auto-Objekt zu definieren, also eine entsprechende Klasse zu programmieren:

```
package{
    public class Auto{
        var Typ:String;
        var Gewicht:Number;
        var PS:Number;
        public function Auto(typ:String, gewicht:Number,
            ps:Number){
            this.PS=ps;
            this.Gewicht=gewicht;
            this.Typ=typ;
        }
        public function zeigeWerte(){
            trace("Typ: "+this.Typ+", Gewicht: "+this.Gewicht+",
                PS: "+this.PS);
        }
    }
}
```

Die Klasse `Auto` verfügt über drei Eigenschaften: `Typ`, `Gewicht` und `PS`. Im Konstruktor werden diese über Konstruktorparameter gesetzt. Außerdem gibt es eine Funktion, die diese Werte ausgibt. Nun steht es Ihnen frei, beliebig viele Objekte dieser Klasse zu erzeugen:

```
var auto:Auto = new Auto("X500", 4, 150);
auto.zeigeWerte();//Ausgabe: Typ: X500, Gewicht: 4, PS: 150
var auto2:Auto = new Auto("X1000", 4, 200);
auto2.zeigeWerte();//Ausgabe: Typ: X1000, Gewicht: 4, PS: 200
```

Wenn Sie dieses Beispiel ausprobieren wollen, beachten Sie, dass es sich um ActionScript 3 handelt. Sie müssen also ein ActionScript 3-Dokument erzeugen und obiges Skript auf Bild 1 der Hauptzeitleiste legen. Die Klasse kommt in eine Datei `Auto.As`, die sich im gleichen Verzeichnis befinden muss.

Vererbung

Stellen Sie sich vor, Sie wollen eine neue Klasse programmieren, haben aber bereits eine Klasse auf Lager, die dieselbe Kernfunktionalität beinhaltet. Nun werden Sie sich wünschen, diese Klasse ganz einfach zu erweitern, ohne alles neu programmieren zu müssen. Auch hierfür stellt die objektorientierte Programmierung ein sehr mächtiges Werkzeug bereit – die Vererbung. Angenommen, Sie möchten Ihre Auto-Anwendung um LKWs erweitern. Diese verfügen über dieselben Eigenschaften wie ein Auto, zusätzlich gibt es aber noch ein Maximalgewicht für die Ladung. Über das Schlüsselwort `extends` können Sie einfach sämtliche Funktionalität der Auto-Klasse übernehmen:

```
package{
    public class Lkw extends Auto{
        var MaxGewicht:Number;
        public function Lkw(typ:String, gewicht:Number, ps:Number,
            maxGewicht:Number){
            super(typ, gewicht, ps);
            this.MaxGewicht = maxGewicht;
        }
        public function zeigeMaxGewicht(){
            trace("Ladegewicht: "+this.MaxGewicht);
        }
    }
}
```

Diese Klasse verfügt über eine neue Eigenschaft, `MaxGewicht`. Dieser wird im Konstruktor der entsprechende Parameter zugeordnet, die restlichen Parameter werden einfach per `super()` an den Konstruktor der übergeordneten Klasse »weitergegeben«. Dann wird eine neue Funktion `zeigeMaxGewicht()` definiert, um das Maximalgewicht anzuzeigen. Nun können Sie Objekte der Klasse erzeugen:

```
var lkw:Lkw = new Lkw("LKW1000", 4, 400, 1000);  
lkw.zeigeWerte();//Ausgabe: Typ: LKW1000, Gewicht: 4, PS: 400  
lkw.zeigeMaxGewicht();//Ausgabe: Ladegewicht: 1000
```

Sie können die von der Klasse `Auto` vererbte Methode `zeigeWerte()` verwenden, obwohl es sie in der Klasse `Lkw` gar nicht gibt. Sämtliche Funktionalität der `Auto`-Klasse ist aber in der `Lkw`-Klasse enthalten. Natürlich können Sie auch die Klasse `Lkw` selbst wieder erweitern. Auf diese Weise können Sie weitverzweigte Strukturen erstellen. Auch `Flash` selbst ist in solch einer Struktur organisiert.

Packages (Pakete)

Wenn Sie sehr viele Klassen haben, ergibt sich irgendwann die Notwendigkeit, diese noch weiter zu strukturieren und zu Packages (Paketen) zusammenzufassen. In `ActionScript 2` ist die Verwendung von Packages optional. In `ActionScript 3` hingegen müssen alle Klassen in einem Package untergebracht sein, was durch das Schlüsselwort `package` geschieht. Wenn Sie jedoch gezielt ein Package erstellen, beispielsweise mit nützlichen Zeichen-Funktionen, die Sie immer wieder benutzen wollen, ist es sinnvoll, dem Package einen Namen zu geben. Dies geschieht folgendermaßen:

```
package testpaket{  
    //Klassendefinitionen  
}
```

Dieser Code erstellt ein Package mit dem Namen `testpaket`. Die Datei muss sich in einem Verzeichnis befinden, das genauso heißt wie die Datei. Unter diesem Namen können Sie es dann in Ihre Projekte importieren:

```
import testpaket.*;
```

Diese Import-Anweisung, beispielsweise in einer Dokumentklasse, stellt die Klassen des Packages `testpaket` zur Verfügung.

In `ActionScript 2` ist das ein wenig anders organisiert. Anstelle des Schlüsselwortes `package` ist hier die Ordnerstruktur in Verbindung mit dem Klassennamen kennzeichnend für Packages. Im Gegensatz zu `ActionScript 3`, wo mehrere Klassen in ein und derselben Datei untergebracht werden können, ist in `ActionScript 2` nur eine Klasse pro Datei zulässig. Klassen, die sich etwa in einem Package `testpaket` befinden sollen, müssen dann

im Verzeichnis `testpaket` liegen und diesen Namen auch in der Klassendefinition tragen:

```
class testpaket.Testklasse{  
    //Code  
}
```

Packages dienen aber nicht nur zur Strukturierung von Klassen. Mit ihrer Hilfe können sogenannte Namespace-Konflikte vermieden werden. Namespaces sind ein komplexes Thema. Grob gesagt sind sie trennbare Bereiche für Namen. Innerhalb eines Namespaces kann jeder Name nur einmal vorkommen. In verschiedenen Namespaces aber kann ein und derselbe Name durchaus mehrfach vorkommen. So können Sie beispielsweise zwei verschiedene Klassen mit dem Namen »Button« erstellen und diese in jeweils eigenen Packages unterbringen. Die eigentlichen Namen Ihrer beiden Button-Klassen sind dann etwa »paketeins.Button« und »paketzwei.Button«, wenn diese in den Packages »paketeins« und »paketzwei« liegen. Man nennt diesen Namen auch den **vollständig qualifizierten Namen**. Wenn Sie eine Klasse importiert haben, können Sie auch den unqualifizierten Namen verwenden:

```
var btn:Button = new Button();
```

Wenn Sie in demselben Projekt die Packages »paketeins« und »paketzwei« importieren und dann über den unqualifizierten Namen Button-Instanzen erstellen wollen, führt das zu einem Namespace-Konflikt. Sie können ihn vermeiden, indem Sie die vollständig qualifizierten Namen verwenden:

```
var btn1:paketeins.Button = new paketeins.Button();  
var btn2:paketzwei.Button = new paketzwei.Button();
```

Eigene Packages und Namespaces kommen hauptsächlich in sehr umfangreichen Projekten zum Einsatz. Für kleinere Projekte genügt es in der Regel, die bereits vorhandenen Packages zu nutzen, die bereits eine Vielzahl von Funktionalitäten zur Verfügung stellen.

Schrift-Effekt

Schrift wie von Hand zeichnen lassen

Ein kleiner, aber netter Effekt ist das Zeichnen lassen eines Schriftzuges »wie von Hand«. In diesem Workshop erstellen Sie mit Hilfe einer Pfadanimation einen solchen Effekt. Dabei kommt eine einfache Zeitleistenanimationen und – ein bisschen – ActionScript zum Einsatz.

Zielsetzungen:

Erstellen einer Pfadanimation

Nachzeichnen der Bewegung mit ActionScript

[Ergebnis: 01_SCHRIFT-EFFEKT • HANDSCHREIB-EFFEKT.SWF]

```
font.moveTo(this.pen._x, this.pen._y);  
font.lineStyle(5, 0x000066, 100);  
function onEnterFrame(){  
    font.lineTo(this.pen._x, this.pen._y);  
}
```



Schritt 1 Vorbereitungen

Im ersten Schritt legen Sie einen neuen Ordner auf Ihrer Festplatte an und nennen ihn SCHRIFT-EFFEKT. Hier werden Sie die von Ihnen erstellten Beispiele zu diesem Workshop abspeichern.

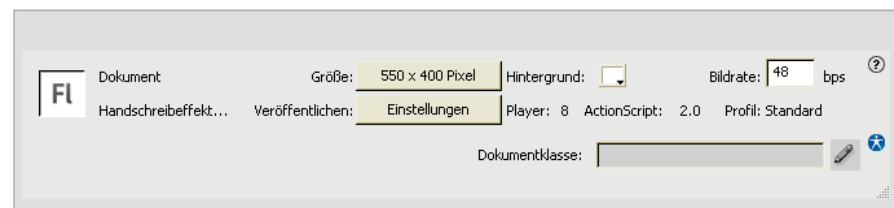
Schritt 2 Anlegen der Flash-Datei

In diesem Workshop spielt sich alles in der Flash-Datei ab. Es wird lediglich ein kleines Skript integriert, das sich ebenfalls in der Flash-Datei befindet. Legen Sie eine neue Flash-Datei an. Wählen Sie eine Bühnengröße von 550x400 Pixel und eine BILDRATE von 48 bps.

Abbildung 1 ►
Filmeinstellungen
vornehmen



▲ Abbildung 2
Fünf Ebenen werden
angelegt.



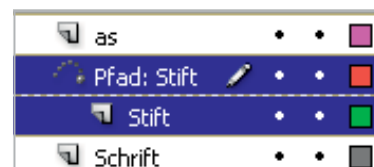
Legen Sie nun fünf Ebenen an, und geben Sie diesen von oben nach unten folgende Namen: »as«, »Stift«, »Schrift«, »Führungsebene« und »Hintergrund«. Importieren Sie außerdem die drei Bilder PAPIERROLLE.JPG, SCHREIBFEDER.PNG und SCHRIFTZUG.JPG in die Bibliothek.

Schritt 3 Ebeneneigenschaften zuweisen

Die Ebenen dienen verschiedenen Zwecken und müssen näher spezifiziert werden. Die Ebene »as« enthält ein kleines Skript. Sie muss nicht verändert werden.

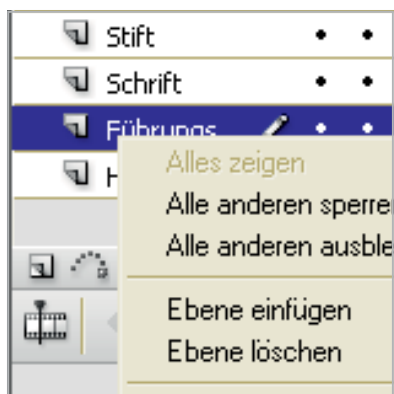
Die Ebene »Stift« wird die Schreibfeder enthalten und benötigt einen Pfad, dem die Feder folgen soll. Klicken Sie daher mit der rechten Maustaste auf die Ebene, und wählen Sie PFAD HINZUFÜGEN.

Abbildung 3 ►
Es erscheint eine weitere
Ebene mit dem Namen
»Pfad: Stift« direkt oberhalb.



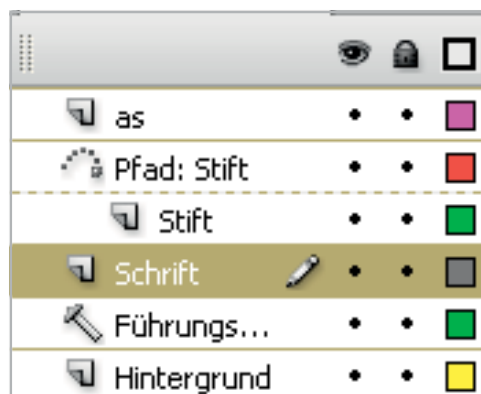
Die Ebenen »Schrift« und »Hintergrund« können unverändert bleiben; die Ebene »Führungsebene« muss erst noch in eine Führungsebene umgewandelt werden. Klicken Sie mit der rechten Maustaste darauf, und wählen Sie FÜHRUNGSEBENE.

Die Führungsebene wird die Vorlage für den Schriftzug, also das Bild SCHRIFTZUG.JPG, enthalten.



▲ **Abbildung 4**

Die Ebene wird in eine Führungsebene konvertiert.



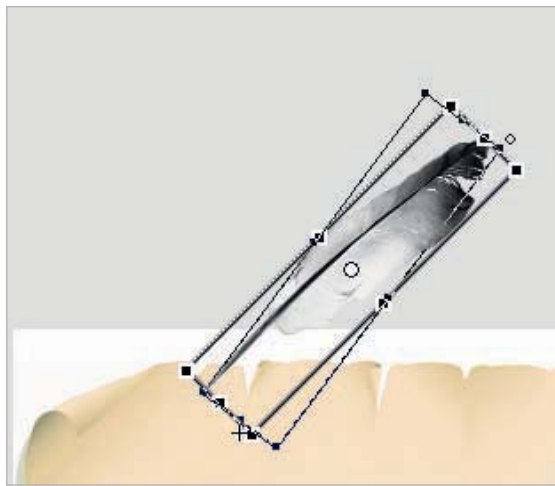
▲ **Abbildung 5**

Der Flash-Film beinhaltet nun bereits recht viele verschiedene Ebenen.

Ziehen Sie nun die Grafik PAPIERROLLE.JPG auf Bild 1 der Ebene »Hintergrund«, und richten Sie sie mittig auf der Bühne aus. Auf Bild 1 der Ebene »Führungsebene« wird die Grafik SCHRIFTZUG.JPG platziert und ebenfalls mittig ausgerichtet.

Ziehen Sie dann die Grafik SCHREIBFEDER.PNG auf Bild 1 der Ebene »Stift«, und konvertieren Sie sie in ein Symbol (Rechtsklick, dann Menüeintrag IN SYMBOL KONVERTIEREN wählen). Wählen Sie als TYP die Option MOVIECLIP, wählen den Mittelpunkt als REGISTRIERUNG und geben einen beliebigen Symbolnamen an. Wählen Sie als Instanznamen im Eigenschaftenfenster »pen« (siehe Abbildung 7 auf der nächsten Seite).

Außerdem können Sie mit dem Frei-Transformieren-Werkzeug die Feder skalieren und ihr eine leichte Neigung geben, wenn Sie möchten.



Schritt 4

Grafiken einfügen

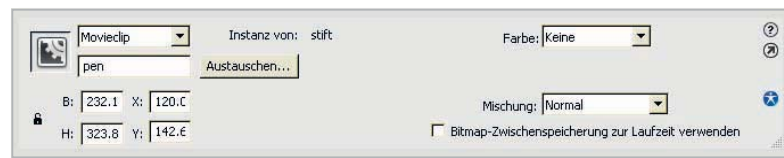
Tipp

Führungsebenen können Sie für einfache Zeitleistenanimationen als Hilfsmittel verwenden. Sie werden im fertigen Flash-Film nicht angezeigt und können z. B. grafische Vorlagen enthalten.

◀ **Abbildung 6**

Die Feder wird leicht geneigt, um die Haltung beim Schreiben anzudeuten.

Abbildung 7 ►
Eigenschaften des Stift-Movieclips



Schritt 5 Pfadbene anlegen

Nun zeichnen Sie den Pfad, an dem sich die Animation entlangbewegt. Klicken Sie auf Bild 1 der Ebene »Pfad: Stift«. Am besten sperren Sie alle anderen Ebenen, was sich übrigens generell empfiehlt, wenn man in einer Ebene zeichnen möchte. Klicken Sie dazu auf das kleine Schloss-Symbol ganz oben. Anschließend entsperren Sie die Pfadebenen wieder, indem Sie auf deren Schloss-Symbol klicken. Wählen Sie nun das Stiftwerkzeug . Zeichnen Sie den Schriftzug der Führungsebene nach.

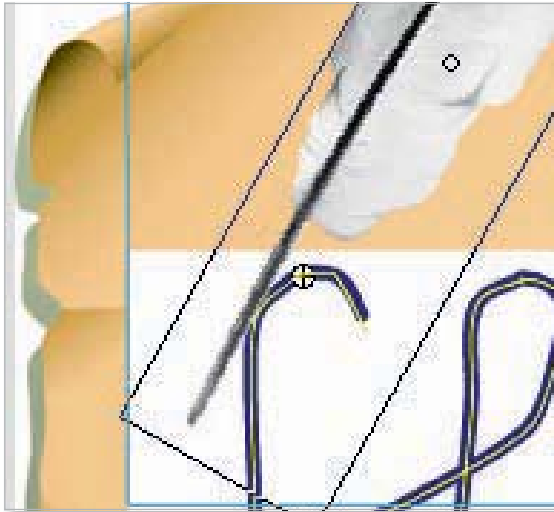
Abbildung 8 ►
Nachzeichnen des Pfades mit dem Stiftwerkzeug



Instanzname und Verknüpfung

Instanznamen sind die Namen von Objekten, die sich in einem Flash-Projekt befinden. Sie können mit ActionScript angesprochen werden. Wird kein Instanzname festgelegt, vergibt Flash Standard-Instanznamen: `instance1`, `instance2` usw. Verwechseln Sie diese nicht mit den Bezeichnern, die über VERKNÜPFUNG im Kontextmenü eines Bibliotheksymbols vergeben werden! Letztere können zwar auch mit ActionScript bis Version 2 angesprochen werden, repräsentieren jedoch das Symbol abstrakt als »Schablone« und nicht die konkrete Instanz.

Fügen Sie anschließend in Bild 600 der Hauptzeitleiste ein Schlüsselbild in der Ebene »Stift« und ein einfaches Bild in allen anderen Ebenen ein. Legen Sie in Bild 600 der Ebene »Stift« eine [stop](#)-Anweisung. Erstellen Sie einen Bewegungs-Tween zwischen Bild 1 und Bild 600, indem Sie mit der rechten Maustaste auf ein beliebiges dazwischenliegendes Bild klicken und BEWEGUNGS-TWEEN ERSTELLEN wählen. Verschieben Sie den Stift-Movieclip in Bild 1 nun so, dass der Registrierpunkt auf dem Anfang des Pfades liegt, und in Bild 600 so, dass der Registrierpunkt auf dem Ende des Pfades liegt. Den Registrierpunkt erkennen Sie an einem kleinen Kreuz. Da der Pfad der Ebene »Stift« zugeordnet ist, rastet der Registrierpunkt am Pfad ein, sobald Sie ihn in die Nähe bewegen.



◀ **Abbildung 9**
Movieclip am Pfad
ausrichten

Testen Sie den Film über `[Strg]/[⌘] + [↶]`. Der Stift bewegt sich entlang des Pfades.

Die Schreibfeder folgt jetzt dem Pfad. Mit ActionScript »verfolgen« wir später diese Bewegung und zeichnen sie nach. Hierbei ist es wichtig, dass sich der Punkt des Stift-Movieclips, der dem Pfad folgt, mit der Spitze der Schreibfeder deckt (denn entlang dieses Punktes wird später nachgezeichnet). Falls dies nicht der Fall ist, müssen Sie entsprechend anpassen. Markieren Sie also eines der beiden Schlüsselbilder in der Ebene »Stift«. Öffnen Sie dann den Stift-Movieclip per Doppelklick, und verschieben Sie die Grafik so, dass die Spitze der Feder den Startpunkt des Pfades berührt.



Schritt 6

Stift-Movieclip anpassen

◀ **Abbildung 10**
Im ersten Bild der Pfad-
animation wird die Schreib-
feder-Grafik so verschoben,
dass die Spitze den Pfad
berührt.

Schritt 7

Nachzeichnen der Bewegung mit ActionScript

Die Bewegung der Schreibfeder ist fertig; das eigentliche Schreiben fehlt aber noch. Hierzu benötigen Sie einen leeren Movieclip, in den nachher der Schriftzug gezeichnet wird. Erstellen Sie einen solchen in der Bibliothek durch Klicken auf das Plus-Symbol am unteren Rand.

Ziehen Sie den leeren Movieclip auf Bild 1 der Ebene »Schrift«, und positionieren Sie ihn genau in der linken oberen Ecke der Bühne.

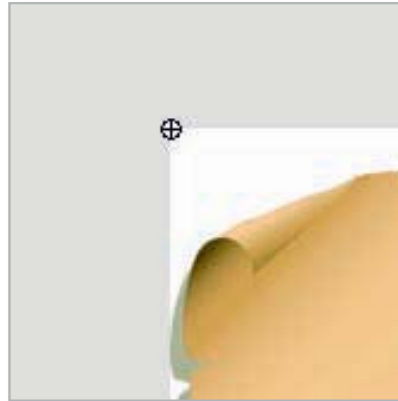


Abbildung 11 ►

Ein leerer Movieclip in der linken oberen Ecke der Bühne

Geben Sie ihm den Instanznamen »font«. Markieren Sie nun Bild 1 der Ebene »as«, und öffnen Sie das Skriptfenster über **F9** bzw. **Alt + F9**. Geben Sie folgenden Code ein:

```
1: font.moveTo(this.pen._x, this.pen._y);
2: font.lineStyle(5, 0x000066, 100);
3: function onEnterFrame(){
4:     font.lineTo(this.pen._x, this.pen._y);
5: }
```

In Zeile 1 wird die aktuelle Zeichnungsposition auf die Position der Schreibfederspitze gesetzt. Dann wird in der Funktion `onEnterFrame()` sukzessive der Schriftzug gezeichnet.

Zu guter Letzt kann ein wenig Feintuning nicht schaden – die Schrift soll schließlich wie von Menschenhand gezeichnet aussehen. Hierzu gibt es die Möglichkeit, den Tween zu bearbeiten. Ein entsprechender Button wird im Eigenschaftenfenster angezeigt, wenn Sie ein beliebiges Bild innerhalb des Tweens markieren.

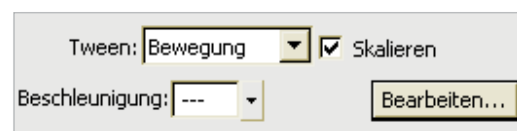
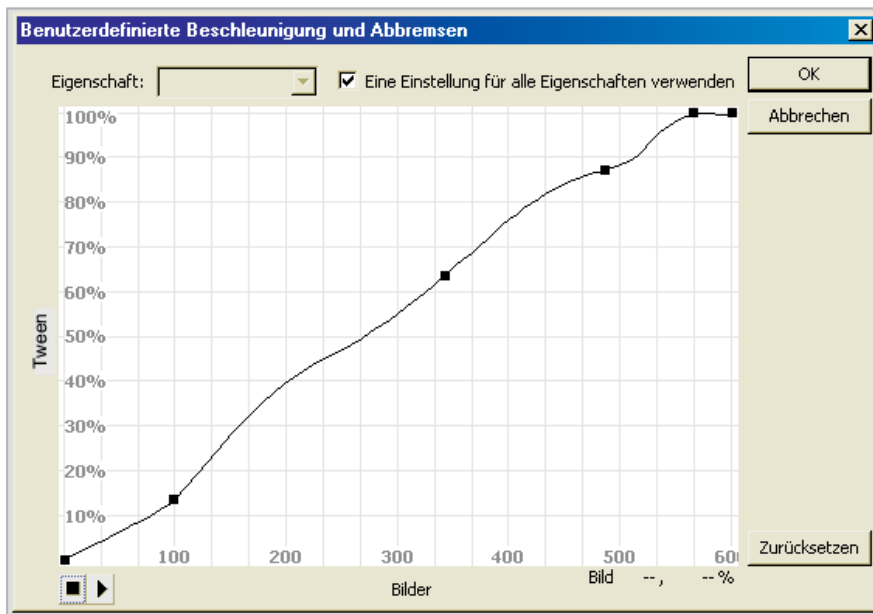


Abbildung 12 ►

Tween bearbeiten

Es erscheint ein Graph, an dem Sie das Feintuning vornehmen können.



◀ **Abbildung 13**
Feintuning der
Stiftbewegung

Experimentieren Sie hier ruhig ein wenig herum, bis Ihnen das Ergebnis gefällt.



◀ **Abbildung 14**
Der Effekt ist fertig!

Schreibmaschinen-Effekt

Einen klassischen Effekt in einer Klasse realisieren

```
class Typewriter{
    var typeSheet:TextField;
    var textToType:String;
    var letters:Array;
    var letterCounter:Number;
    var typeInterval:Number;
    var speed:Number;
    public function Typewriter(textToType:String,
    typeSheet:TextField){
        this.typeSheet = typeSheet;
        this.letterCounter = 0;
        this.textToType = textToType;
    }
    public function go(speed:Number, varname:String):Void{
        this.typeSheet.text = "";
        this.speed = speed||200;
        typeText(this.textToType);
    }
    private function typeText(textToSplit:String):Void{
        this.letters = textToSplit.split("");
        this.typeInterval = setInterval(this, "typeLetter", this.
        speed);
    }
    private function typeLetter():Void{
        this.typeSheet.text += this.letters[this.letterCounter];
        this.letterCounter++;
        if (letterCounter == this.letters.length){
            clearInterval(this.typeInterval);
            this.letterCounter = 0;
        }
    }
}
```

Das Erscheinen eines Textes Buchstabe für Buchstabe ist ein Klassiker unter den Effekten und in Flash über die Zeitleiste relativ einfach zu realisieren. In diesem Workshop gehen Sie einen kleinen Schritt weiter und erstellen eine ActionScript 2-Klasse, die Sie fortan immer wieder für diesen Zweck benutzen können.

Zielsetzungen:

ActionScript 2-Klasse erzeugen

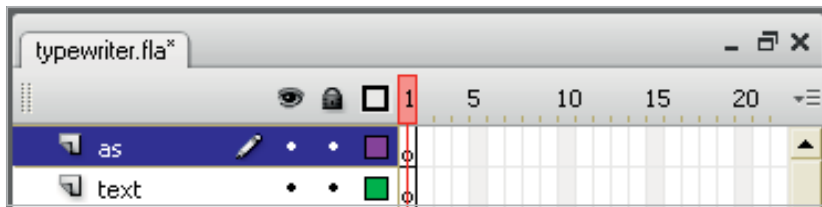
Schreibmaschinen-Effekt für eine Zeichenkette erstellen

[Ergebnis: 02_SCHREIBMASCHINEN-EFFEKT • TYPEWRITER.SWF]

Hallo ich bin ein Text

Erstellen Sie eine neue Flash-Datei (ActionScript 2). Da diese nur Testzwecken dient, können Sie die Standardeinstellungen übernehmen. Auch die Bildrate ist unerheblich, da der Schreibmaschinen-Effekt intervallgesteuert ist.

Legen Sie in der neuen Datei zwei Ebenen an, von denen die eine das Textfeld zur Textausgabe, die andere nur den ActionScript-Code enthalten wird.

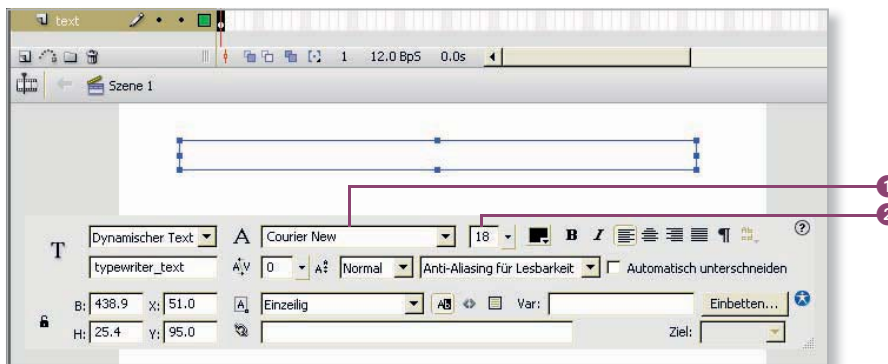


Schritt 1

Anlegen der Flash-Datei

◀ **Abbildung 1**
Zwei neue Ebenen »as« und »text« werden angelegt.

Legen Sie in der Ebene »text« ein dynamisches Textfeld an. Geben Sie diesem im Eigenschaftenfenster den Instanznamen »typewriter_text«. Stellen Sie als Schrift **COURIER NEW** ❶ und als Schriftgröße 18 Punkt ❷ ein. Dieses Feld dient später zur Darstellung des Textes.



◀ **Abbildung 2**
Dynamisches Textfeld

Erstellen Sie in demselben Verzeichnis, in dem auch Ihre Flash-Datei liegt, eine ActionScript-Datei. Wählen Sie dazu DATEI • NEU und anschließend im Reiter ALLGEMEIN den Eintrag ACTIONSCRIPT-DATEI. Speichern Sie die Datei unter dem Namen TYPEWRITER.AS ab.

Zunächst legen Sie die Klasse an und deklarieren die Klassenvariablen:

```
class Typewriter{
    var typeSheet:TextField;
    var textToType:String;
    var letters:Array;
```

Schritt 2

Anlegen der ActionScript-Datei

Deklarieren & Initialisieren

In ActionScript unterscheidet man zwischen der Deklaration und der Initialisierung von Variablen. Die Initialisierung ist das Zuweisen eines Wertes. Davon unabhängig wird durch die Deklaration ganz schlicht bekanntgegeben, dass eine Variable eines bestimmten Typs und Namens existiert.

```
var letterCounter:Number;  
var typeInterval:Number;  
var speed:Number;  
}
```

Es gibt insgesamt sechs Klassenvariablen: Das Textfeld, in dem der Text dargestellt wird (`typeSheet`), den darzustellenden Text (`textToType`), ein Array, in das die einzelnen Buchstaben von `textToType` geschrieben werden, um sie danach einzeln auslesen zu können. Die Variable `letterCounter` wird benötigt, um die Elemente des Arrays `letters` durchzuzählen. `typeInterval` ist die ID des Intervalls, das die einzelnen Buchstaben ausgibt. Schließlich gibt es noch eine Variable `speed`, die die Anzahl von Millisekunden angibt, die zwischen dem Erscheinen zweier Buchstaben liegen.

Schritt 3 Der Konstruktor

Der Konstruktor initialisiert drei Klassenvariablen und empfängt den Instanznamen des Textfeldes und den Ausgabertext. Fügen Sie an das Ende Ihrer Klasse, direkt vor der schließenden Klammer, folgenden Code ein:

```
1: public function Typewriter(textToType:String,  
   typeSheet:TextField){  
2:     this.typeSheet = typeSheet;  
3:     this.letterCounter = 0;  
4:     this.textToType = textToType;  
5: }
```

Konstruktor

Ein Konstruktor ist eine Funktion, die ausgeführt wird, sobald eine neue Instanz einer Klasse erzeugt wird.

Der Konstruktor-Parameter `textToType` beinhaltet den Text, der durch den Typewriter ausgegeben werden soll.

In den Zeilen 2 bis 4 werden die übergebenen Parameter den Klassenvariablen zugeordnet und die Variable `letterCounter` auf 0 gesetzt.

Schritt 4 Erstellen der Kernfunktionalität

In diesem Schritt verarbeiten Sie den übergebenen Text. Das heißt, Sie zerlegen den Text in ein Array mit den einzelnen Buchstaben und starten dann ein Intervall, das den Inhalt des Arrays Schritt für Schritt ausgibt. Außerdem erstellen Sie eine sogenannte öffentliche (`public`) Funktion, die die Ausgabe startet. Denn im Konstruktor selbst werden nur Variablen initialisiert. Durch dessen bloßen Aufruf passiert noch nichts.

Fangen wir von vorn an – mit der öffentlichen Funktion, die die Textausgabe startet. Fügen Sie wieder ganz an das Ende Ihrer Klasse folgenden Code ein:

```

1: public function go(speed:Number):Void{
2:     this.typeSheet.text = "";
3:     this.speed = speed||200;
4:     typeText(this.textToType);
5: }

```

Die Funktion empfängt den Parameter `speed` (vergleiche Schritt 2) und weist ihn der Klassenvariablen `speed` zu. Ist er nicht definiert, wird er in Zeile 3 auf 200 gesetzt.

Außerdem wird der Inhalt des Textfeldes – sofern vorhanden – in Zeile 2 gelöscht, und eine Funktion `typeText` wird aufgerufen. Diese sorgt für die Ausgabe des Textes Buchstabe für Buchstabe. Hierfür wird ein Intervall genutzt. Fügen Sie unterhalb der Funktion `go()` folgenden Code ein:

```

1: private function typeText(textToSplit:String){
2:     this.letters = textToSplit.split("");
3:     this.typeInterval = setInterval(this, "typeLetter", this.
    speed);
4: }

```

Die Funktion `typeText` macht aus dem Ausgabertext ein Array (`letters`). Hierzu wird die String-Funktion `split()` verwendet, die einen String in ein Array zerlegt. Hierfür kann ein Trennzeichen angegeben werden. Dieses wird beim Zerlegen entfernt und markiert die Stelle, an der zerlegt wird. So würde etwa ein Komma, also `»,«,` den String `»a,b,c«` in ein Array mit den Elementen `»a«, »b«` und `»c«` zerlegen. Die Angabe eines leeren Strings (`""`) als Trennzeichen erzeugt ein Array mit allen einzelnen Zeichen – genau das, was wir wollen.

Zeile 3 startet ein Intervall, das gemäß dem Wert von `speed` eine Funktion wiederholt ausführt und dessen ID der Klassenvariablen `typeInterval` zugeordnet wird.

Die Funktion `typeLetter` gibt einen einzelnen Buchstaben aus und wird im nächsten Schritt beschrieben.

Die Ausgabe der einzelnen Buchstaben ist zwar nicht schwierig, aber trotzdem sozusagen das Herzstück der Klasse. Ergänzen Sie folgenden Code direkt unter der Funktion `typeText`:

```

1: private function typeLetter():Void{
2:     this.typeSheet.text += this.letters[this.letterCounter];
3:     this.letterCounter++;
4:     if (letterCounter == this.letters.length){

```

Video-Training

Was es mit Strings genau auf sich hat und wie Sie mit diesem Datentyp in ActionScript umgehen, erfahren Sie auch in der Video-Lektion 2.2 auf der Buch-DVD.

Schritt 5

Ausgabe einzelner Buchstaben


```

5:     clearInterval(this.typeInterval);
6:     this.letterCounter = 0;
7:   }
8: }

```

Klassenvariablen

Klassenvariablen sind innerhalb der Klasse global und können aus jeder Funktion heraus angesprochen werden, während lokale Variablen, die etwa innerhalb einer Funktion deklariert werden, auch nur innerhalb dieser Funktion gültig sind.

In Zeile 2 wird dem Text des Textfeldes, das zu Anfang leer ist, jeweils ein Element des Arrays `letters` hinzugefügt. Als Index wird die Klassenvariable `letterCounter` verwendet. `letterCounter` wird in Zeile 3 hochgezählt. So wird sukzessive der komplette Text ausgegeben. In Zeile 4 wird überprüft, ob das Ende des Arrays erreicht ist. Wenn dem so ist, wird das Intervall gelöscht.

Da Intervalle über IDs angesprochen werden und Sie als ID eine Klassenvariable verwendet haben, können Sie das Intervall problemlos aus dieser Funktion heraus ansprechen. Für den Fall, dass die Textausgabe erneut erfolgen soll (erneutes Aufrufen von `go()`), wird `letterCounter` auf 0 zurückgesetzt.

Schritt 6 Anwenden der Klasse

Herzlichen Glückwunsch, die Typewriter-Klasse ist fertig! Zeit, sie in Aktion zu sehen! Wechseln Sie in die Flash-Datei. Legen Sie folgenden Code auf Bild 1 der Hauptzeitleiste in der Ebene »as«:

```

1: var tw:Typewriter = new Typewriter("Hallo ich bin ein Text",
   typewriter_text);
2: tw.go();

```

Der Text »Hallo ich bin ein Text« wird mit der Standardverzögerung von 200 Millisekunden zwischen zwei Zeichen ausgegeben.

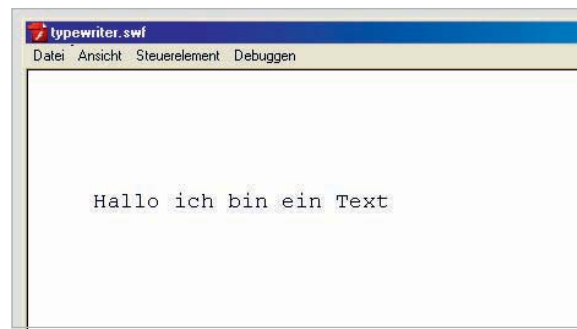


Abbildung 3 ►

Die Typewriter-Klasse funktioniert! Flash gibt den Text Buchstabe für Buchstabe aus.

Um einen anderen Zeitabstand zu erreichen, können Sie diesen natürlich auch explizit mit `tw.go(50);` angeben. Die Ausgabe ist nun viermal so schnell.

Zufallsanimation

Mit Hilfe der Tween-Klasse krabbelt eine Spinne zufallsgesteuert hin und her.

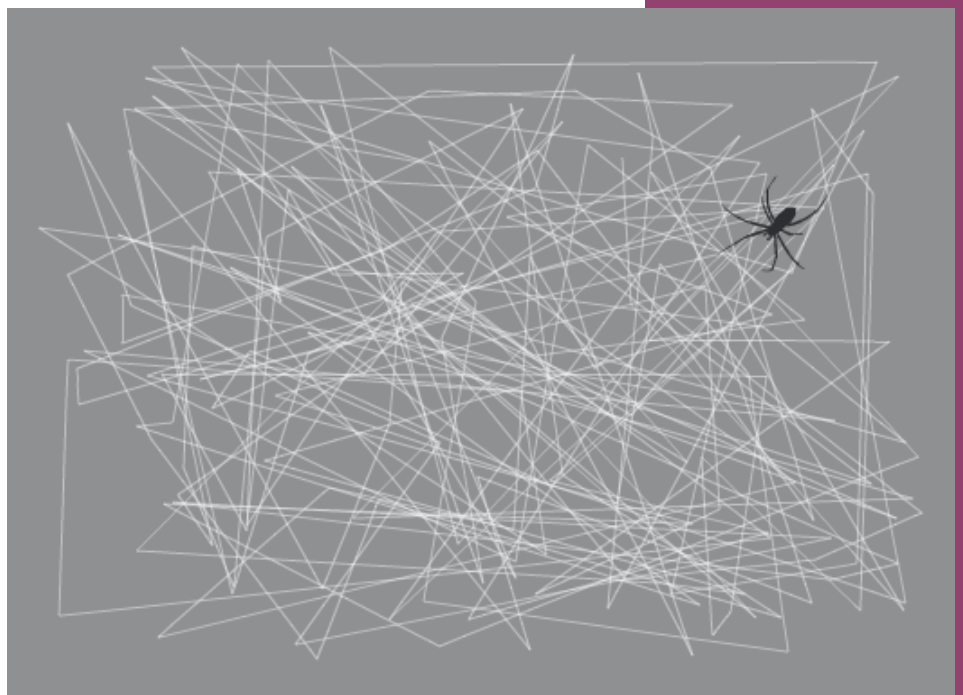
Anstatt wie bisher Movieclips aufwendig über die Zeitleiste zu animieren, sollten Sie zukünftig darüber nachdenken, ob Sie die gleiche Animation nicht einfacher per Skript erledigen könnten. In diesem Workshop lernen Sie den Umgang mit der Flash-eigenen Tween-Klasse kennen.

Zielsetzungen:

Tween-Klasse benutzen
EventListener
implementieren
Animationsloop erstellen

[Ergebnis: 03_ZUFALLS-
ANIMATION • SPINNE.SWF]

```
import mx.transitions.Tween;
import mx.transitions.easing.Regular;
import mx.utils.Delegate;
class Spider extends MovieClip
{
    private var maxX:Number;
    private var maxY:Number;
    private var newX:Number;
    private var newY:Number;
    private var tweenX:Tween;
    private var tweenY:Tween;
    private var tweenRotation:Tween;
    private var lastX:Number;
    private var lastY:Number;
    public function Spider()
    {
        maxX = Stage.width;
        maxY = Stage.height;
        _x = lastX = Math.round(_width/2+Math.random()*
            (maxX-_width));
```



Schritt 1 Vorbereitungen

Im ersten Schritt legen Sie einen neuen Ordner auf Ihrer Festplatte an. Nennen Sie diesen ZUFALLSANIMATION. Hier werden Sie die von Ihnen erstellten Beispiele zu diesem Workshop abspeichern.

Schritt 2 Erstellen der Flash-Datei

Wechseln Sie zu Flash, und erstellen Sie über **Strg**/**⌘**+**N** eine neue FLASH-DATEI (ACTIONSCRIPT 2.0) ❶. Bestätigen Sie den Dialog mit **Ok** ❷.

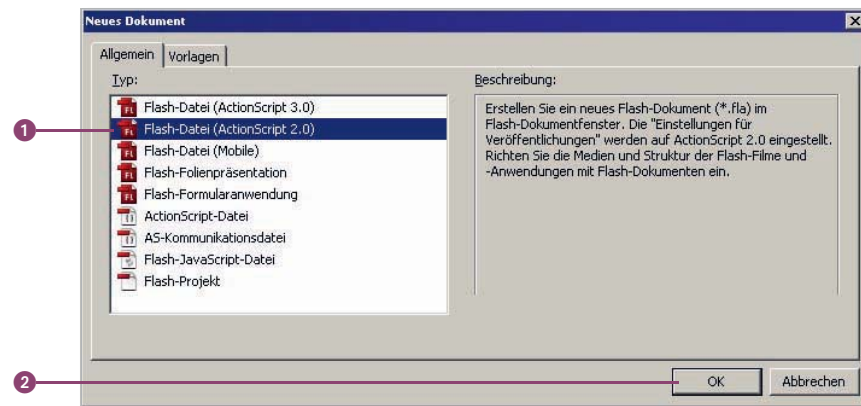


Abbildung 1 ▶
Legen Sie ein neues Flash-Dokument an.

Diese Datei speichern Sie mit **DATEI • SPEICHERN UNTER** als **SPINNE.FLA** in dem in Schritt 1 erstellten Ordner ab.

Schritt 3 Anpassen der Dokument-eigenschaften

Anschließend passen Sie die **DOKUMENTEIGENSCHAFTEN** über **Strg**/**⌘**+**J** an: Die **GRÖSSE** belassen Sie bei den eingestellten **550x400 Pixeln** ❸, ändern jedoch die **HINTERGRUNDFARBE** auf ein mittleres Grau (**#999999**) ❹ und die **BILDRATE** auf **25 bps** ❺. Bestätigen Sie die Änderungen mit **Ok**.

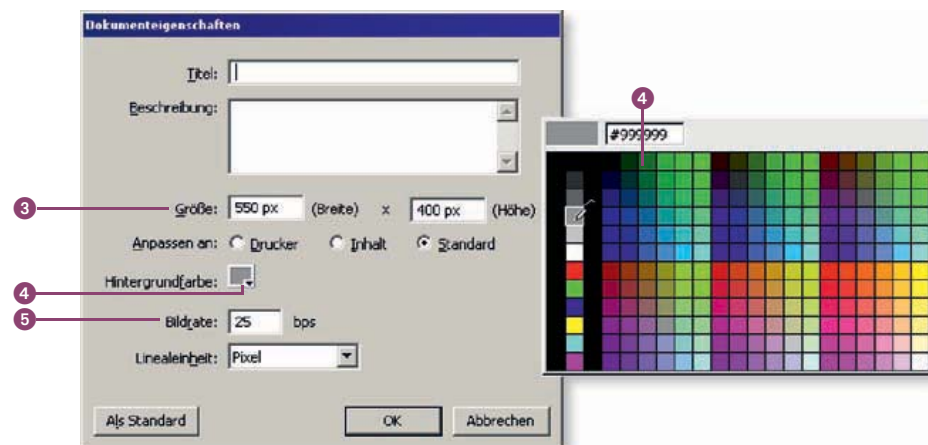
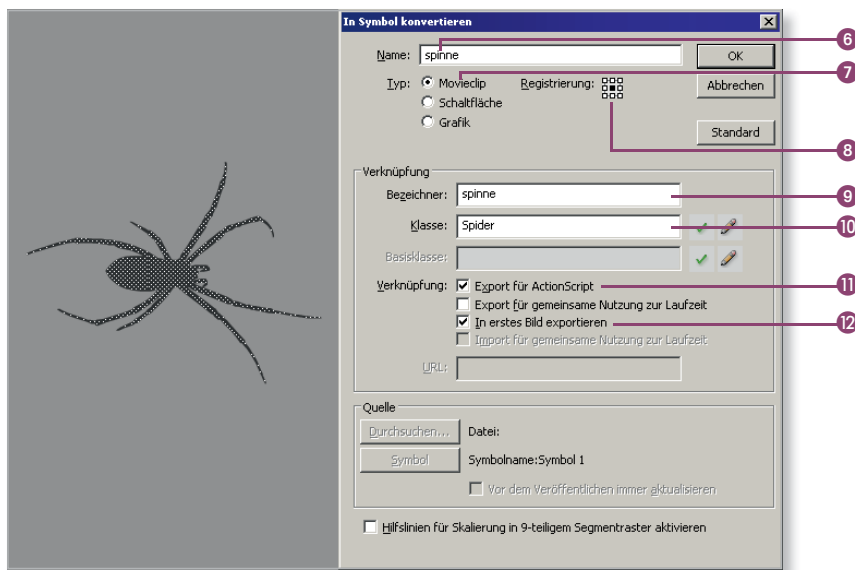


Abbildung 2 ▶
Die Hintergrundfarbe ändern Sie durch einen Klick auf das graue Kästchen neben **HINTERGRUNDFARBE**. Anschließend wählen Sie mit der Pipette eine neue Farbe.

Zeichnen Sie eine Spinne mit Hilfe des Stiftwerkzeugs  oder machen Sie es sich einfacher und kopieren Sie die Spinnen-Grafik aus der fla-Beispieldatei von der Buch-DVD. Wenn Sie beim Erzeugen der Spinne auf die Beispieldatei zurückgreifen, achten Sie darauf, dass Sie nicht den kompletten Movieclip benutzen, sondern nur die Form innerhalb des Spinnen-Grafik-Movieclips. Sie werden im Laufe des Workshops die Form in einen Movieclip konvertieren.

Nach Fertigstellung der Spinnen-Form wählen Sie diese mit **[Strg]/[⌘]+[A]** komplett aus und konvertieren sie über **[F8]** in ein Symbol. Im **IN SYMBOL KONVERTIEREN**-Dialog geben Sie dem Symbol den Namen »spinne« **6**, wählen als **TYP MOVIECLIP** aus **7** und ändern die **REGISTRIERUNG** auf **mittig** **8**. Um die Vorbereitungen abzuschließen, müssen Sie noch die erweiterten Optionen anpassen.

Wenn Sie in der erweiterten Ansicht sind, setzen Sie zuerst einen Haken bei **EXPORT FÜR ACTIONSCRIPT** **11** – es sollte dann automatisch ein zweiter Haken bei **IN ERSTES BILD EXPORTIEREN** erscheinen **12**. Danach tragen Sie als **BEZEICHNER** »spinne« ein **9** und weisen dem Symbol die **KLASSE** »Spider« zu **10**, die Sie im nächsten Schritt noch erzeugen werden. Bestätigen Sie alle Angaben mit **OK**.



Schritt 4

Erzeugen der Spinne

Tipp

Im **IN SYMBOL KONVERTIEREN**-Dialog wechseln Sie über die Schaltfläche **ERWEITERT/STANDARD** zwischen der erweiterten Ansicht und der Standard-Ansicht.

◀ Abbildung 3

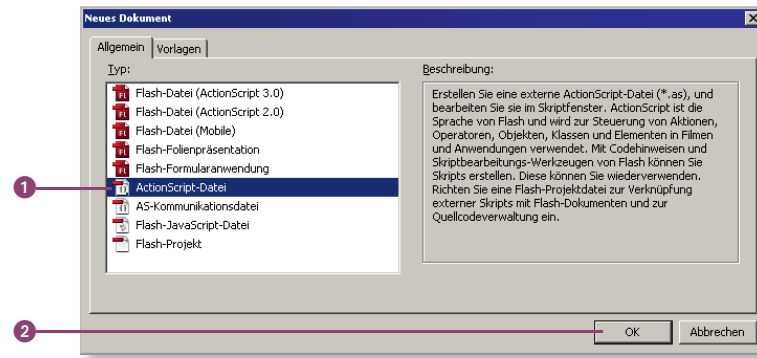
Die Spinne wird in einen Movieclip konvertiert.

Um der Spinne Leben einzuhauchen, haben Sie ihr im vorherigen Schritt eine Klasse zugewiesen. Diese Klasse werden Sie nun erstellen. Zu diesem Zweck öffnen Sie mit **[Strg]/[⌘]+[N]** eine neue **ACTIONSCRIPT-DATEI** **1** (Abbildung 4). Bestätigen Sie den Dialog mit **Ok** **2**.

Schritt 5

Anlegen der Spider-Klasse

Abbildung 4 ►
Eine neue ActionScript-Datei
anlegen



Speichern Sie die as-Datei mit `[Strg]/[⌘] + [S]` unter dem Namen SPIDER.AS im Projektordner ab.

Schritt 6

Erstellen des Klassengerüsts

Legen Sie die Klasse wie folgt an:

```
1: class Spider extends MovieClip
2: {
3:     private var maxX:Number;
4:     private var maxY:Number;
5:     public function Spider()
6:     {
7:         maxX = Stage.width;
8:         maxY = Stage.height;
9:         _x = Math.round(_width/2+Math.random()*
            (maxX-_width));
10:        _y = Math.round(_height/2+Math.random()*
            (maxY-_height));
11:    }
12: }
```

Da die Spinne, der Sie diese Klasse zugewiesen haben, ein Movieclip ist und Sie innerhalb der Klasse auf die Movieclip-spezifischen Eigenschaften der Spinne zugreifen wollen, erbt `Spider` in Zeile 1 von `MovieClip`.

In den Zeilen 3 und 4 deklarieren Sie jeweils eine Variable für die maximale x- und y-Position, die die Spinne einnehmen kann, ohne von der Bühne zu verschwinden. Innerhalb des Konstruktors (Zeilen 7 und 8) initialisieren Sie die soeben erzeugten Variablen mit der Breite und Höhe der Bühne. In den beiden darauffolgenden Zeilen 9 und 10 setzen Sie die x- bzw. y-Position der Spinne auf einen zufälligen Wert. Dieser Wert wird so gewählt, dass die Spinne auf jeden Fall komplett auf der Bühne zu sehen

Video-Training

Was der Konstruktor ist und was er genau macht, erfahren Sie auch in der Video-Lektion 3.2 auf der Buch-DVD.

sein wird. Durch die `Math.round`-Funktion garantieren Sie, dass die Spinne immer auf ganzzahligen Koordinaten liegt.

Speichern Sie die bearbeitete Klasse mit `Strg/⌘+S`; wechseln Sie danach wieder zur `SPINNE.FLA`-Datei, und testen Sie diese über den Tastaturbefehl `Strg/⌘+↵`.



▲ **Abbildung 5**

Zur Verdeutlichung der zufallsgesteuerten Positionierung wurden in dieser Abbildung die unterschiedlichen Positionen der Spinne übereinandergelegt. Diesen Effekt werden Sie in Flash so nicht erzielen.

Die Spinne sollte innerhalb des Flash-Films zufallsgesteuert positioniert werden – drücken Sie im aktiven Flash-Player-Fenster wiederholt `Strg/⌘+↵`. Bei jedem erneuten Laden der Datei sollte die Spinne an einer anderen x- und y-Position auftauchen.

Damit sich die Spinne bewegen kann, müssen Sie zuerst bestimmen, wohin die Spinne »krabbeln« soll. Auch diese Position wird per Zufall festgelegt.

Fügen Sie deshalb nach Zeile 4 der `Spider`-Klasse aus Schritt 6 folgenden Code ein:

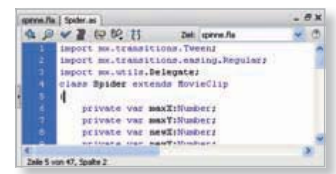
```
private var newX:Number;  
private var newY:Number;
```

Schritt 7

Testen des Flash-Films

Tipp

Wenn nur eine fla-Datei geöffnet ist, können Sie diese direkt aus dem ActionScript-Editor heraus mit `Strg/⌘+↵` testen. Bei mehreren geöffneten fla-Dateien, die getestet werden könnten, gibt es in der rechten oberen Ecke des Editors eine ZIEL-Auswahlliste, in der Sie die fla-Datei bestimmen können, die beim Testen kompiliert werden soll.



▲ **Abbildung 6**
ZIEL-Auswahlliste

Schritt 8

Zielposition der Spinne festlegen

In diesen beiden Zeilen führen Sie zwei neue Klassenvariablen ein, die die neue x- bzw. y-Position speichern werden. Anschließend ergänzen Sie im Konstruktor aus Schritt 6 eine Zeile (Zeile 1), schließen ihn wieder (Zeile 2) und fügen eine neue Funktion ein.

```
1:         chooseNewSpot();
2:     }
3:     private function chooseNewSpot():Void
4:     {
5:         newX = Math.round(_width/2+Math.random()*
6:             (maxX-_width));
7:         newY = Math.round(_height/2+Math.random()*
8:             (maxY-_height));
9:     }
```

Die `chooseNewSpot`-Methode ab Zeile 3 wird aus dem Konstruktor heraus aufgerufen (Zeile 1) und lässt die Spinne sich einen neuen Zielpunkt suchen. Dies geschieht genau so, wie Sie die Startposition innerhalb des Konstruktors bestimmt haben. Die Zuweisung der neuen x- und y-Werte findet in den Zeilen 5 und 6 statt.

Schritt 9

Bewegen Sie die Spinne

Damit sich die Spinne nun vom Startpunkt aus zu dem neu berechneten Zufallspunkt bewegt, benutzen Sie in diesem Schritt die Flash-eigene Tween-Klasse. Diese und die Beschleunigungs-Klasse `Regular` müssen vor Gebrauch in Ihrer Klasse zuerst importiert werden.

```
import mx.transitions.Tween;
import mx.transitions.easing.Regular;
```

Anschließend werden zwei weitere Klassenvariablen deklariert – sie werden später als Container für die beiden Tween-Objekte der x- bzw. y-Position dienen. Ergänzen Sie diese über dem Konstruktor:

```
private var tweenX:Tween;
private var tweenY:Tween;
```

Nun müssen Sie noch die Funktion `chooseNewSpot` um folgende Zeile ergänzen und danach die Funktion `crawlAway` erstellen:

```

1:         crawlAway();
2:     }
3:     private function crawlAway():Void
4:     {
5:         tweenX = new Tween(this, "_x", Regular.easeInOut, _x,
            newX, 3, true);
6:         tweenY = new Tween(this, "_y", Regular.easeInOut, _y,
            newY, 3, true);
7:     }

```

Sie rufen in Zeile 1 nach der Bestimmung der Zielposition die neue `crawlAway`-Methode auf. Diese sorgt ab Zeile 3 dafür, dass die Spinne mit Hilfe von zwei Tweens ihre x- und y-Position verändert.

In Zeile 5 erzeugen Sie ein solches Tween-Objekt. Dieses erwartet als Übergabeparameter neben dem zu bewegendem Movieclip (`this`) noch den Parameter (`_x`), der geändert werden soll, eine Beschleunigungs-Methode (engl.: *easing method*; in diesem Fall `Regular.easeInOut()`), den aktuellen und den neuen Wert (`_x`, `newX`) sowie das Zeit-Intervall 3.

Als Letztes wird ein boolescher Wert angehängt, der angibt, ob das davor übergebene Intervall als Sekunden (`true`) oder Bilder (`false`) zu interpretieren ist. In diesem Fall »tweenen« Sie die x- und y-Position der Spinne innerhalb von drei Sekunden mit der `Regular.easeInOut`-Methode vom Ausgangswert zur Zielposition. »*easeInOut*« heißt nichts anderes, als dass die Beschleunigung am Anfang leicht zunimmt und gegen Ende der Animation stetig wieder abnimmt.

Speichern Sie die Änderungen der `SPIDER.AS`-Datei, und testen Sie mit `Strg`/`⌘` + `↵` den neuen Flash-Film.



Tipp

Die wohl bekanntesten Beschleunigungs-/Easing-Methoden stammen von Robert Penner. Mehr dazu erfahren Sie unter folgenden Links:

- <http://robertpenner.com/easing/>
- http://robertpenner.com/easing/penner_chapter7_tweening.pdf

Schritt 10

Testen des Flash-Films

◀ Abbildung 7

Die Spinne bewegt sich von der Startposition zu einer zufälligen zweiten Position. Danach verharrt sie allerdings an dieser Position.

Schritt 11

EventListener hinzufügen

Damit die Spinne nicht einfach aufhört zu krabbeln, implementieren Sie in diesem Schritt einen `onMotionFinished`-EventListener, der nach dem Beenden der Animation den Ablauf erneut anstößt.

Hierzu fügen Sie in die `crawlAway`-Methode aus dem vorherigen Schritt unter die Erstellung der beiden Tweens folgende Zeile ein:

```
tweenX.onMotionFinished = Delegate.create(this,
    chooseNewSpot);
```

Delegate.create()

Die `Delegate`-Klasse ermöglicht es Ihnen, eine Funktion in einem bestimmten Gültigkeitsbereich auszuführen. In diesem Fall nutzen Sie sie dazu, den Aufruf der `onMotionFinished`-Methode des Tween-Objekts innerhalb Ihrer Spider-Klasse zu verarbeiten.

Per `Delegate` verweisen Sie den Aufruf der `onMotionFinished`-Methode des x-Tweens an die bereits bestehende Methode `chooseNewSpot` innerhalb Ihrer Klasse. Somit sucht sich die Spinne nach Erreichen des Zielpunktes gleich wieder einen neuen Punkt auf der Bühne.

Damit Sie die `Delegate`-Klasse benutzen können, müssen Sie diese am Anfang Ihrer Klasse importieren:

```
import mx.utils.Delegate;
```

Speichern Sie die geänderte Klasse mit `[Strg]/[⌘]+[S]` ab.

Schritt 12

Flash-Film erneut testen

Wenn Sie nun den Flash-Film mit `[Strg]/[⌘]+[↵]` testen, werden Sie feststellen, dass die Spinne von einem Punkt zum nächsten krabbelt und nicht mehr aufhört. Aber leider schaut sie nicht in die Richtung, in die sie krabbelt und wirkt deshalb sehr unrealistisch.

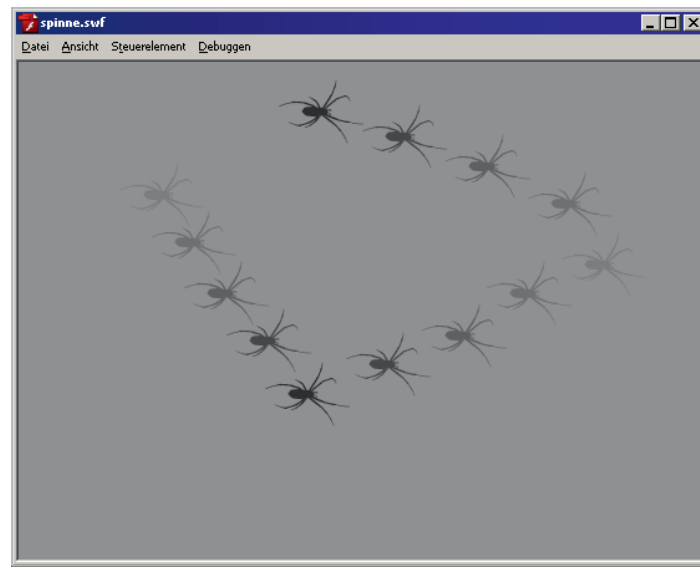


Abbildung 8 ►
Die Spinne krabbelt fleißig
über den Bildschirm.

Um die korrekte Drehung der Spinne zu bestimmen, müssen Sie den Winkel ausrechnen, in dem die Spinne sich von dem einen Punkt zum nächsten bewegt. Dies ist mit Hilfe der `Math.atan2`-Funktion innerhalb von Flash sehr leicht.

Die Berechnung des Winkels fügen Sie in die `chooseNewSpot`-Methode ein, nachdem Sie den neuen Zielpunkt der Spinne bestimmt haben. Da die Rotation der Spinne nicht einfach so angepasst werden soll, werden Sie, genau wie bei der x- und y-Koordinate, einen Tween zu Hilfe nehmen.

Deklarieren Sie oberhalb des Konstruktors mit den anderen Klassenvariablen folgende Variable:

```
private var tweenRotation:Tween;
```

Hier nun die überarbeitete Methode:

```
1: private function chooseNewSpot():Void
2: {
3:     newX = Math.round(_width/2+Math.random()*(maxX-_width));
4:     newY = Math.round(_height/2+Math.random()*(maxY-_height));
5:     var deltaX:Number = newX-_x;
6:     var deltaY:Number = newY-_y;
7:     var degree:Number = Math.atan2(deltaY, deltaX)*
        180/Math.PI;
8:     tweenRotation = new Tween(this, "_rotation",
        Regular.easeInOut, _rotation, degree, 1.5, true);
9:     tweenRotation.onMotionFinished = Delegate.create(this,
        crawlAway);
10: }
```

In den Zeilen 5 und 6 berechnen Sie den x- und y-Abstand zwischen dem alten und dem neuen Punkt. Diese Werte benötigen Sie zur weiteren Verwendung in Zeile 7.

Über die vorher bestimmten Abstände berechnen Sie den Winkel zwischen den beiden Punkten im Bogenmaß – die Multiplikation am Ende der Zeile überträgt den Wert aus dem Bogenmaß in eine Gradzahl. Diesen Wert können Sie als `_rotation` für die Spinne benutzen.

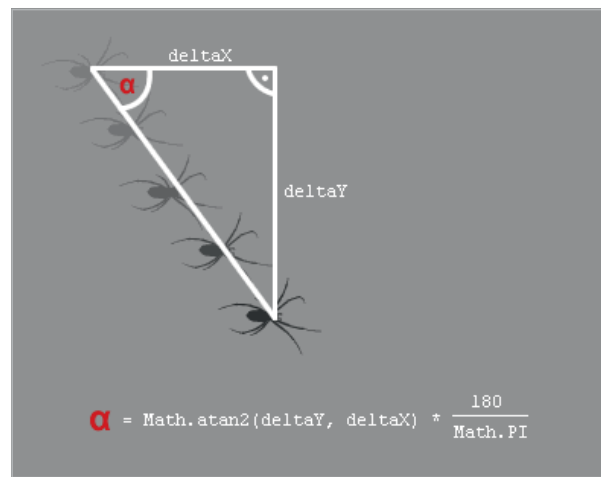
Schritt 13

Rotation der Spinne an die Laufrichtung anpassen

Math.atan2()

Die `Math.atan2`-Funktion berechnet den Winkel Alpha zwischen zwei Punkten eines rechtwinkligen Dreiecks (siehe Abbildung 9). Sie liefert das Ergebnis im Bogenmaß zurück. Um diesen Wert in eine Gradzahl umzurechnen, müssen Sie das Resultat mit 180 multiplizieren und anschließend durch Pi dividieren.

Abbildung 9 ►
So berechnen Sie den Winkel, der die Basis für die Rotation der Spinne bildet.



Zeile 8 erzeugt den Rotations-Tween: Die Spinne soll anhand der `Regular.easeInOut`-Funktion innerhalb von 1,5 Sekunden vom jetzigen Rotationswert auf den soeben neu berechneten Wert (degree) gedreht werden.

Der direkte Aufruf der `crawlAway`-Methode fällt zugunsten des neuen `onMotionFinished`-EventListeners in Zeile 9 weg.

Speichern Sie die Klasse wiederum mit `Strg`/`⌘` + `S` ab.

Schritt 14 Testen des Flash-Films

Nachdem Sie den Flash-Film über `Strg`/`⌘` + `↵` neu erstellt haben, werden Sie sehen, dass die Spinne nun immer mit dem Kopf voran über die Bühne krabbelt.

Lediglich die Beine der Spinne sind noch etwas steif – sie gleitet doch eher, als dass sie krabbelt. Dem werden Sie im nächsten Schritt Abhilfe verschaffen.

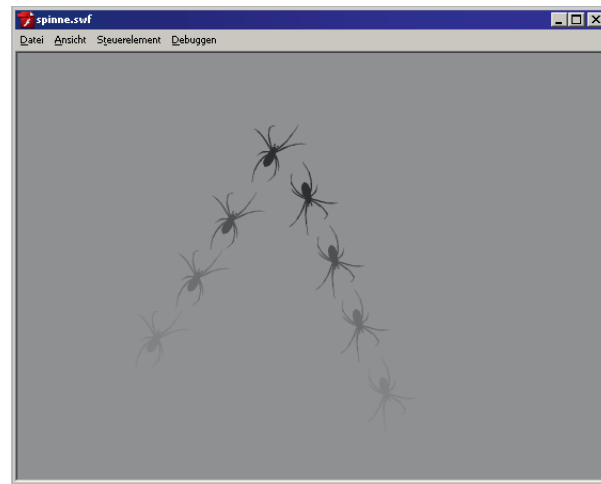
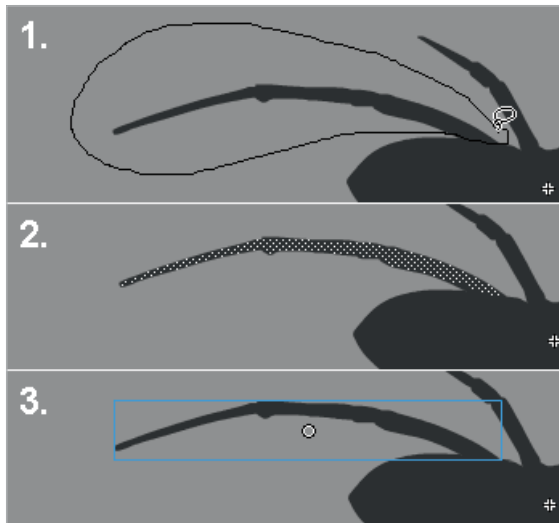


Abbildung 10 ►
Die Spinne schaut immer in die Richtung, in die sie krabbelt.

Wählen Sie innerhalb des Spinnen-Movieclips mit Hilfe des Lassowerkzeugs  die einzelnen Beine der Spinne aus, und konvertieren Sie diese mit **[F8]** Bein für Bein zu jeweils einem neuen Movieclip. Benennen Sie die Movieclips mit »bein1« bis »bein8«.

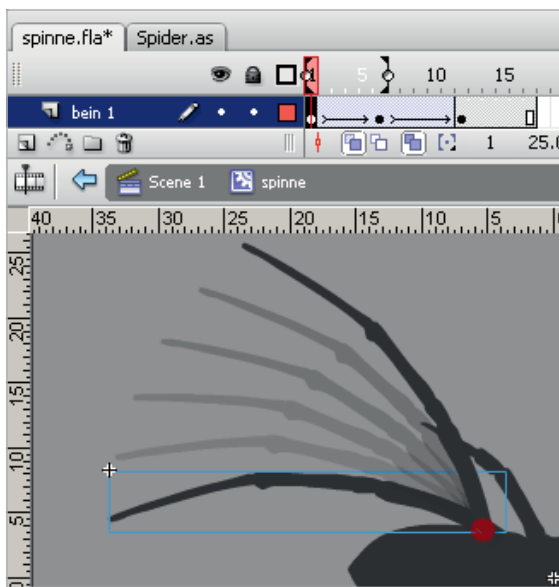


Schritt 15

Die Spinnenbeine animieren

◀ **Abbildung 11**
Jedes Spinnenbein wird in einen Movieclip umgewandelt.

Nachdem Sie die Movieclips erstellt haben, können Sie diese auf der Zeitachse mit einfachen Bewegungs-Tweens animieren. Hierbei ist es nur wichtig zu beachten, dass Sie die Verbindungsstelle zwischen Körper und Bein als Drehpunkt nehmen (roter Kreis). Um dies zu erreichen, müssen Sie den Kreis, der in der vorherigen Abbildung zu sehen ist, mit der Maus auf die in der nächsten Abbildung gekennzeichnete Stelle ziehen.



◀ **Abbildung 12**
Einfache Bewegungs-Tweenings machen das Krabbeln der Spinne realistischer.

Bevor Sie nun den Flash-Film ein letztes Mal testen, werden Sie noch das Spinnennetz, das die Spinne spinnt, während sie von einer Ecke in die andere krabbelt, integrieren.

Schritt 16

Spinnen Sie das Netz

Damit die Spinne auch wirklich ein Netz spinnt, müssen Sie noch drei kleine Änderungen an der Spider-Klasse vornehmen. Als Erstes benötigen Sie zwei weitere Klassenvariablen – diese speichern die Werte der letzten x- bzw. y-Position der Spinne. Ergänzen Sie beide direkt über dem Konstruktor:

```
private var lastX:Number;
private var lastY:Number;
```

Den Konstruktor erweitern Sie dahin gehend, dass die Anfangskoordinaten nicht allein gesetzt, sondern auch in den neuen Variablen gespeichert werden:

```
public function Spider()
{
    maxX = Stage.width;
    maxY = Stage.height;
    _x = lastX = Math.round(_width/2+Math.random()*
(maxX-_width));
    _y = lastY = Math.round(_height/2+Math.random()*
(maxY-_height));
    chooseNewSpot();
}
```

Jetzt fehlt lediglich noch eine `onEnterFrame`-Methode, die in jedem Bild (engl.: frame) eine Linie von der letzten zur aktuellen Position der Spinne zeichnet. Ergänzen Sie folgende Zeilen ganz am Ende Ihrer Klasse:

```
1: private function onEnterFrame():Void
2: {
3:     _root.lineStyle(1, 0xffffffff, 50);
4:     _root.moveTo(lastX, lastY);
5:     _root.lineTo(_x, _y);
6:     lastX = _x;
7:     lastY = _y;
8: }
```

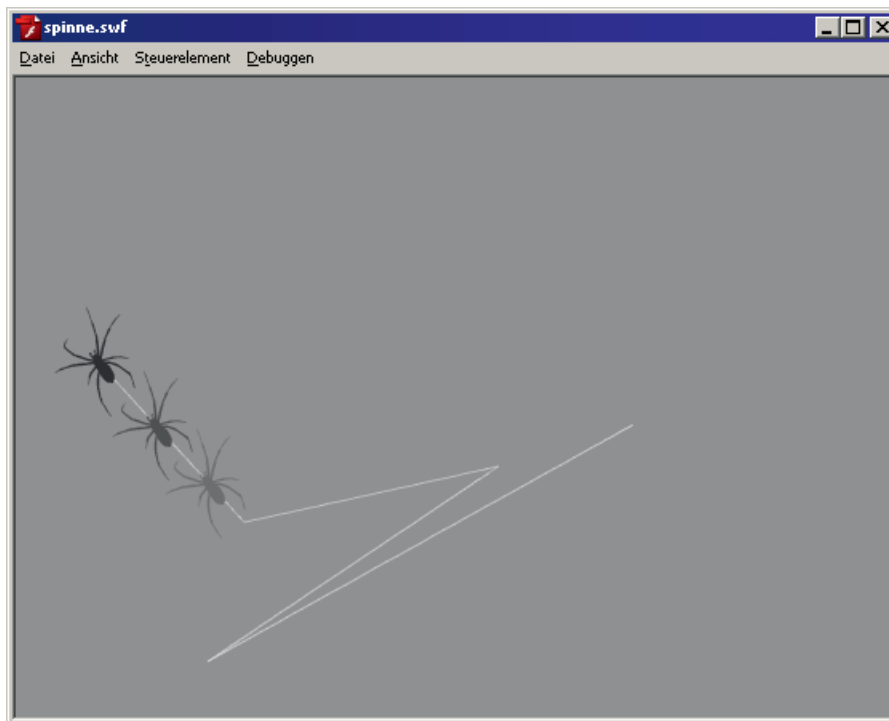
In Zeile 3 legen Sie fest, wie die Linie aussehen soll, die gezeichnet wird. Dabei bestimmt der erste Parameter die Linienstärke in Pixel, der zweite die Farbe und der dritte die Transparenz.

In der darauffolgenden Zeile wird per `moveTo`-Funktion der Startpunkt der zu zeichnenden Linie angesprungen. Gezeichnet wird in Zeile 5 – die Linie verläuft dann von `lastX/lastY` bis `_x/_y`.

In den beiden Zeilen 6 und 7 wird die aktuelle Position der Spinne zwischengespeichert, so dass nicht immer der ganze Faden neu gezeichnet werden muss, sondern nur das Stück, das seit dem letzten Bild dazugekommen ist.

Speichern Sie die Änderungen mit `Strg/⌘+S`.

Wenn Sie nun den Flash-Film erneut über `Strg/⌘+↵` testen, werden Sie sehen, dass die Spinne einen transparenten weißen Faden hinter sich herzieht. Und sie bewegt ab sofort beim Krabbeln auch ihre Beine.



Schritt 17

Flash-Film testen

◀ **Abbildung 13**
Der fertige Flash-Film – ganz schön eklig! ;-)

Scroll-Effekt

Setzen Sie Flash-Filter für eine weichgezeichnete Scrollbewegung ein.

```
package
{
    import flash.display.MovieClip;
    import flash.events.Event;
    import flash.filters.BlurFilter;
    public class Main extends MovieClip
    {
        private var blur:BlurFilter;
        private var targetY:Number = 0;
        public function Main()
        {
            content.mask = smask;
            blur = new BlurFilter(0, 0);
            addEventListener(Event.ENTER_FRAME, onEnterFrame);
        }
    }
}
```

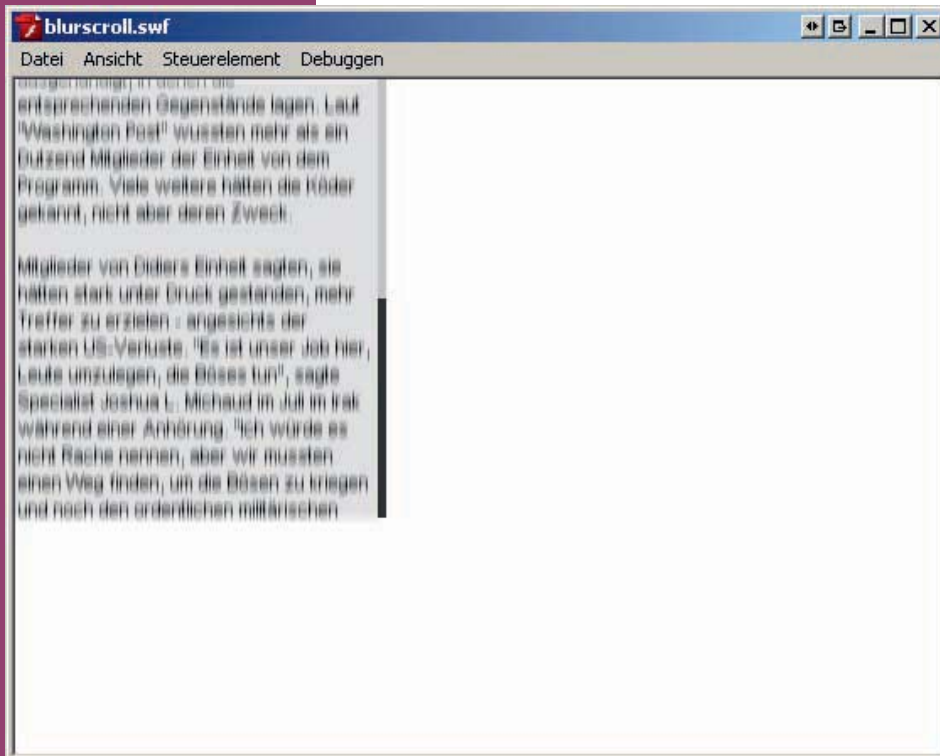
Der weichgezeichnete Scroller war einer der ersten Effekte, der nach der Einführung der Filter in Flash populär wurde. Er kann also mit Fug und Recht als Klassiker bezeichnet werden. In diesem Workshop lernen Sie, diesen Effekt mit wenigen Codezeilen selbst zu erzeugen.

Zielsetzungen:

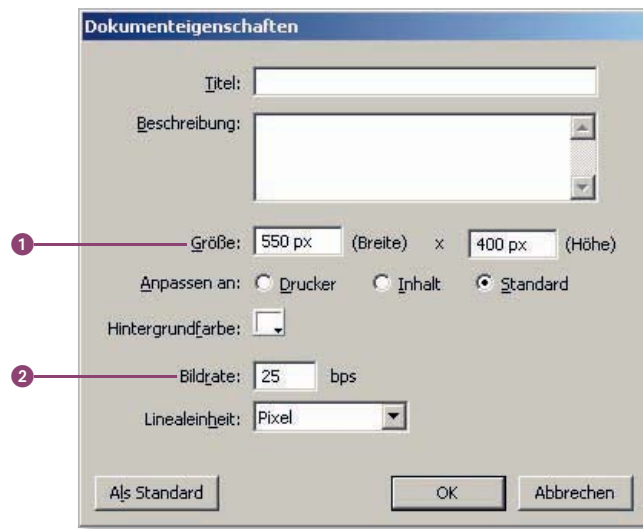
- Masken erstellen
- Funktionsfähigen Scroller programmieren
- Flash-Filter einsetzen

[Ergebnis: 04_SCROLL-EFFEKT

• BLURSCROLL.SWF]



Erstellen Sie ein neues AS3-Flash-Dokument mit der Grösse 550x400 Pixel **1** und einer BILDRATE von 25 bps **2**. Speichern Sie dieses Dokument unter dem Namen BLURSCROLL.FLA ab.



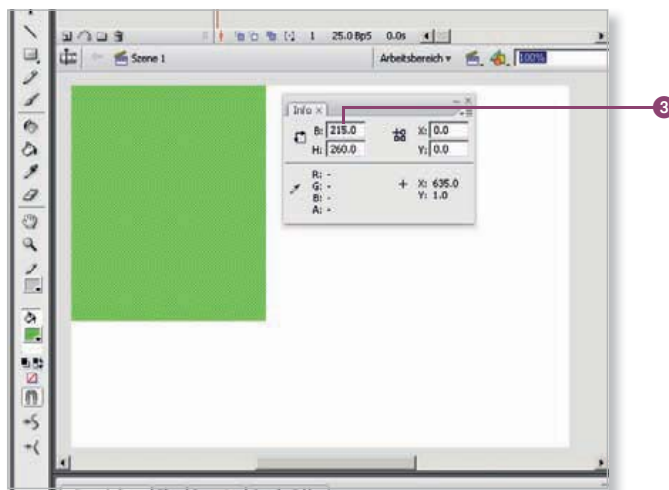
Schritt 1

Dokument erstellen

◀ Abbildung 1

Die Dokumenteigenschaften erreichen Sie durch einen Klick auf den Button neben GRÖSSE im Eigenschaftenfenster.

Erstellen Sie mit dem Rechteckwerkzeug  ein 215x260 Pixel großes Quadrat. Die Maße können Sie in der INFO-Palette **3** eintragen. Da Sie diese Fläche als Maske verwenden werden, spielen die Rahmen- und Füllfarbe keine Rolle.



Schritt 2

Maske erstellen

◀ Abbildung 2

Das Rechteck wird später im Flash-Film nicht zu sehen sein. Es dient lediglich als Maske.

Konvertieren Sie diese Fläche anschließend mit MODIFIZIEREN • IN SYMBOL KONVERTIEREN... in einen Movieclip, und geben Sie diesem den Namen »mask« **4**. Bestätigen Sie mit OK.

Wenn eine Maske per ActionScript angewandt werden soll (was später noch der Fall sein wird), muss diese Maske als Movieclip vorliegen. Eine einfache Fläche reicht dafür nicht aus.

Geben Sie diesem Movieclip nun im Eigenschaftenfenster den Instanznamen »smask« **5**.

Die Ebene, auf der das Rechteck liegt, benennen Sie in »mask« um.

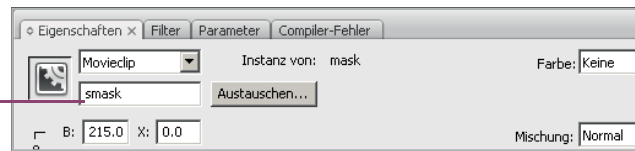
Abbildung 3 ►

Konvertieren Sie die Maske in einen Movieclip.

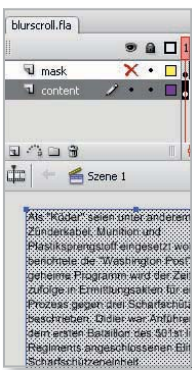


Abbildung 4 ►

Geben Sie dem Movieclip den Instanznamen »smask«.



Schritt 3 Movieclip für den gescrollten Inhalt



▲ Abbildung 5

Auf der Ebene »content« wird ein grau hinterlegtes Textfeld angelegt.

Abbildung 6 ►

Konvertieren Sie Text und Fläche in einen Movieclip.

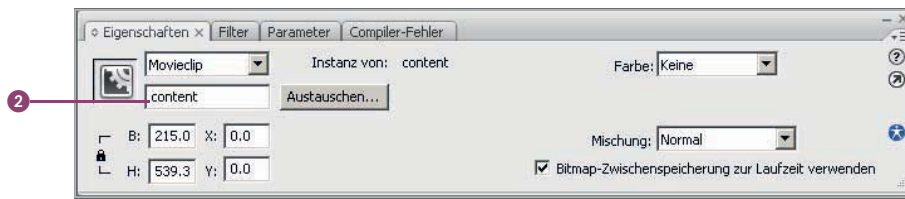
Nun erstellen Sie den Movieclip, der später gescrollt werden soll. Fügen Sie dazu zunächst eine neue Ebene ein, die Sie »content« nennen. Legen Sie mit Hilfe des Rechteckwerkzeugs  eine Fläche mit einer Breite von 215 und einer Höhe von 600 Pixel an. Geben Sie dieser Fläche die Farbe Grau.

Nun platzieren Sie auf diesem grauen Rechteck mit dem Textwerkzeug  einen beliebigen Text. Achten Sie darauf, dass die Grenzen der Fläche nicht überschritten werden.

Markieren Sie nun sowohl die graue Fläche als auch das Textfeld, und konvertieren Sie alles über MODIFIZIEREN • IN SYMBOL KONVERTIEREN... in einen Movieclip. Im nun folgenden Dialogfenster vergeben Sie als NAME »content« **1** und bestätigen mit OK.

Als Instanznamen vergeben Sie anschließend im Eigenschaftenfenster ebenfalls »content« **2**.



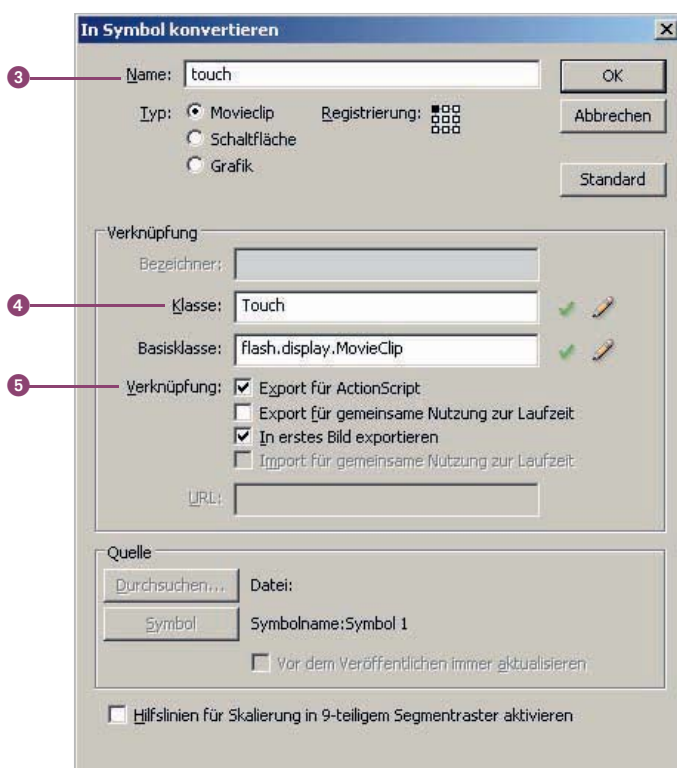


◀ **Abbildung 7**

Geben Sie dem Movieclip den Instanznamen »content«.

Nachdem Sie eine weitere Ebene mit dem Namen »scroll« hinzugefügt haben, erstellen Sie auf dieser Ebene mit dem Rechteckwerkzeug  eine zweite Fläche. Diese Fläche hat eine Größe von 5x130 Pixel und sollte schwarz sein.

Über MODIFIZIEREN • IN SYMBOL KONVERTIEREN... erstellen Sie daraus anschließend einen Movieclip. Im Dialogfenster vergeben Sie dann den Namen »touch« **3** und setzen das Häkchen bei EXPORT FÜR ACTIONSCRIPT **5**. Klappen Sie dazu durch einen Klick auf den Button ERWEITERT/STANDARD die Ansicht mit den erweiterten Optionen auf. Weisen Sie Ihrem Movieclip die KLASSE »Touch« zu **4**, und bestätigen Sie mit OK.



Schritt 4

Scrollbalken anlegen

TIPP

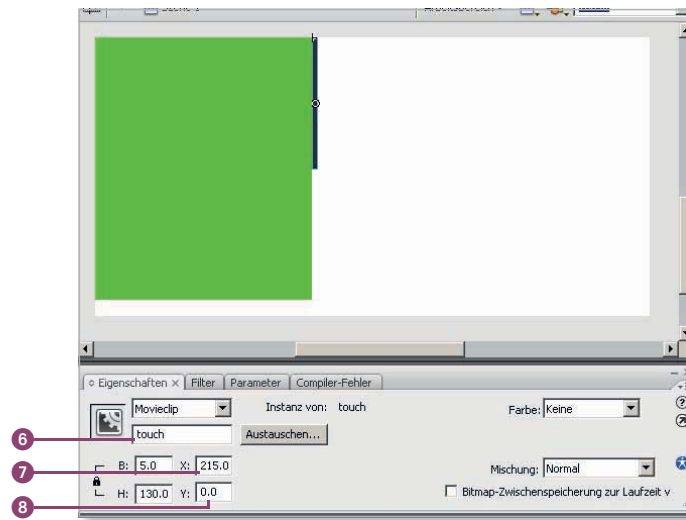
Namen von Symbolen dienen lediglich der Orientierung im Flash-Projekt und haben keine technische Bedeutung. Sie sind nicht zu verwechseln mit Bezeichnern, die von ActionScript bis Version 2 interpretiert werden können. Dennoch empfiehlt es sich, immer sinnvolle Namen für Symbole zu vergeben – spätestens bei einem größeren Projekt verlieren Sie sonst den Überblick.

◀ **Abbildung 8**

Konvertieren Sie den Scrollbalken in einen Movieclip, und binden Sie die Klasse »Touch« an diesen Movieclip.

Danach müssen Sie im Eigenschaftenfenster den Instanznamen »touch« **6** vergeben.

Abbildung 9 ►
Geben Sie dem Scrollbalken
den Instanznamen »touch«
und positionieren Sie ihn
neben dem Inhaltsbereich.



Nun können Sie den Movieclip noch an die richtige Stelle positionieren. Am einfachsten geht das im Eigenschaftenfenster: Die x-Koordinate ist 215 **7**, die y-Koordinate ist 0 **8**.

Schritt 5 Anfasser-Klasse für die Scrollbalken

Erstellen Sie über DATEI • NEU • ACTIONSCRIPT-DATEI eine neue ActionScript-Datei, und speichern Sie diese unter dem Namen TOUCH.AS in demselben Ordner wie die fla-Datei ab. Geben Sie folgenden Code ein:

```
1: package
2: {
3:     import flash.display.MovieClip;
4:     import flash.events.MouseEvent;
5:     import flash.geom.Rectangle;
6:     public class Touch extends MovieClip
7:     {
8:         public function Touch()
9:         {
10:             addEventListener(MouseEvent.MOUSE_DOWN,
11:                 onMouseDown);
12:             addEventListener(MouseEvent.MOUSE_UP, onMouseUp);
13:             stage.addEventListener(MouseEvent.MOUSE_UP,
14:                 onMouseUp);
15:         }
16:         private function onMouseDown(evt:MouseEvent):void
17:         {
18:             this.startDrag(false, new Rectangle(215, 0, 0,
19:                 130));
20:         }
21:     }
22: }
```

```

17:      }
18:      private function onMouseUp(evt:MouseEvent):void
19:      {
20:          this.stopDrag();
21:      }
22:  }
23: }

```

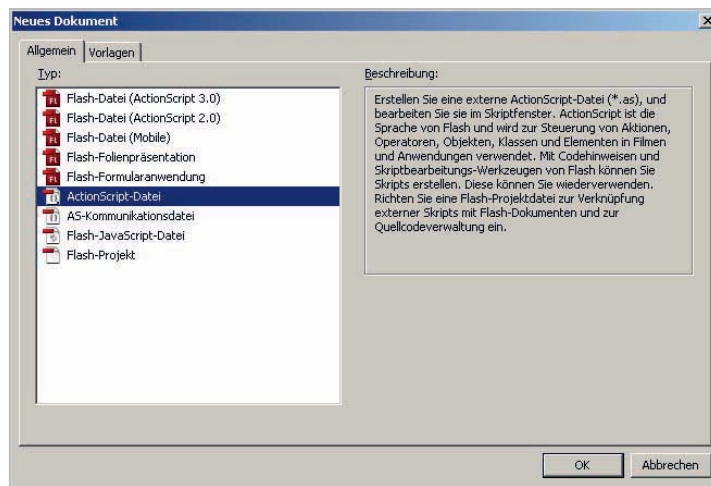
Die Touch-Klasse kontrolliert die Bewegung des Scrollbalkens. In den Zeilen 10 bis 12 meldet sich das Skript bei den Mouse-Events an.

Wenn der User die Maus über dem Anfasser drückt, wird der Scrollbalken in Zeile 16 mit der `startDrag`-Methode bewegt. Das übergebene Rectangle-Objekt legt fest, dass sich der Anfasser vertikal auf der y-Position in einer Spanne von 0 bis 130 bewegen darf.

In Zeile 20 wird, wenn der User die Maus loslässt, der Drag-Vorgang mit `stopDrag` beendet.

Wenn Sie Ihren Flash-Film nun testen, können Sie den Scrollbalken bereits bewegen. Der Content bewegt sich allerdings noch nicht.

Erstellen Sie über **DATEI • NEU • ACTIONSCRIPT-DATEI** eine weitere Klasse, die Sie als »Main.as« in demselben Ordner wie die fla-Datei abspeichern. Diese Klasse wird das Bewegen und Weichzeichnen des `content-Movieclips` übernehmen.



Anschließend geben Sie die Klasse im Feld **DOKUMENTKLASSE** der Datei **BLURSCROLL.FLA** an.

Video-Training

In ActionScript 3 ist einiges anders. Einen Überblick über die wichtigsten Neuerungen können Sie sich in den Video-Lektion 1.1, »Neuerungen in ActionScript 3«, und 1.2, »Das neue Ereignismodell«, auf der Buch-DVD verschaffen.

Schritt 6 Dokumentklasse anlegen

◀ **Abbildung 10**
Neue ActionScript-Datei erstellen

Dokumentklasse

Die Dokumentklasse wird auf der Hauptzeitleiste des Flash-Films ausgeführt. In AS2 musste man Initialisierungscode immer in den ersten Frame der Hauptzeitleiste schreiben. Seit AS3 kann man dies bequem in einer Dokumentklasse machen.

Abbildung 11 ►

Die neu erstellte Action-Script-Datei wird in der fla-Datei als Dokumentklasse angegeben.



Schritt 7

Klassengrundgerüst

Zwischenergebnis:
04_SCROLL-EFFEKT •
MAIN_STEP07.AS



Fügen Sie der eben erstellten Klasse folgenden Code hinzu:

```
1: package
2: {
3:     import flash.display.MovieClip;
4:     import flash.events.Event;
5:     import flash.filters.BlurFilter;
6:     public class Main extends MovieClip
7:     {
8:         private var blur:BlurFilter;
9:         private var targetY:Number = 0;
10:        public function Main()
11:        {
12:        }
13:    }
14: }
```

In den Zeilen 3 bis 5 importieren Sie zunächst die benötigten Klassen.

In Zeile 9 deklarieren Sie die Variable `targetY`, die die Zielposition von `content` speichert.

Schritt 8

Content-Movieclip maskieren

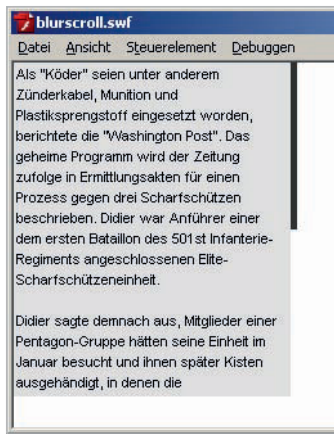
Zwischenergebnis:
04_SCROLL-EFFEKT •
MAIN_STEP08.AS



Fügen Sie Ihrem Skript innerhalb der Funktion `Main()` folgende Zeile hinzu:

```
content.mask = smask;
```

In dieser Zeile legen Sie fest, dass der Movieclip `content` von dem Movieclip `smask` maskiert werden soll. So ist von dem Movieclip `content` immer nur die Fläche der Maske sichtbar. Testen Sie dies, indem Sie den Flash-Film nun veröffentlichen.



◀ Abbildung 12

Der Movieclip content wird mit der Maske smask maskiert.

Um den Content-Movieclip `content` bei der Bewegung des Scrollbalkens weichzuzeichnen, benötigen Sie ein `BlurFilter`-Objekt.

Fügen Sie in der `Main`-Funktion folgende Zeile in Ihre Klasse ein:

```
blur = new BlurFilter(0, 0);
```

Hier setzen Sie den Filter zunächst auf 0, später werden Sie die Werte ändern.

Für die Animation müssen Sie Ihr Skript zunächst bei dem `ENTER_FRAME`-Event anmelden. Ergänzen Sie die `Main`-Funktion um folgende Zeile:

```
addEventListener(Event.ENTER_FRAME, onEnterFrame);
```

Von nun an wird in jedem Frame Ihres Flash-Films die Funktion `onEnterFrame` aufgerufen. Diese Funktion muss natürlich noch erstellt werden.

Jetzt werden Sie die `onEnterFrame`-Funktion erstellen, die für die Bewegung des Content-Movieclips zuständig ist.

Fügen Sie folgenden Code unter die Funktion `Main()` ein:

```
1: private function onEnterFrame(evt:Event):void
2: {
3:     var perc:Number = touch.y / 130;
4:     targetY = (content.height - 260) * -perc;
5:     var dif:Number = (targetY - content.y)/5;
```

Schritt 9

Weichzeichnungsfilter erstellen



Zwischenergebnis:
04_SCROLL-EFFEKT •
MAIN_STEP09.AS

Schritt 10

ENTER_FRAME anmelden



Zwischenergebnis:
04_SCROLL-EFFEKT •
MAIN_STEP10.AS

Schritt 11

Animationsskript erstellen



Zwischenergebnis:
04_SCROLL-EFFEKT •
MAIN_STEP11.AS

```
6:     content.y += dif;
7:     blur.blurY = dif > 0 ? dif : dif*-1;
8:     content.filters = [blur];
9: }
```

In Zeile 3 errechnen Sie aus der y-Position des Scrollanfassers eine Variable, die die Position des Scrollbalkens in Prozent angibt, um die neue Position von `content` festlegen zu können.

Die Zielvariable `targetY` errechnen Sie in Zeile 4. Dabei wird die Höhe von `content` minus der Maskenhöhe (also des sichtbaren Bereichs), mal der zuvor errechneten Prozentzahl genommen.

Um den Inhaltsmovieclip `content` in einer weichen Bewegung zu animieren, wird in Zeile 5 die temporäre Variable `diff` erstellt. Dabei wird die Zielposition `targetY` minus der aktuellen y-Position von `content`, geteilt durch 5 genommen.

Die Y-Position von `content` wird in Zeile 6 mit dem Wert `dif` addiert. So nähert sich die y-Position von `content` immer mehr dem Wert von `targetY` an, bis sie schließlich 0 ist.

In Zeile 7 setzen Sie dann die vertikale Weichzeichnung des `BlurFilter`-Objekts auf den Wert von `dif`. Dem `BlurFilter`-Objekt müssen positive Werte übergeben werden. Da `dif` aber auch negativ sein kann, wird mit der `if`-Abfrage sichergestellt, dass, wenn `dif` negativ ist, `dif` mit `-1` multipliziert wird. Je stärker die Bewegung (also die Distanz, die gescrollt werden muss) ist, desto höher ist der Wert der Variablen `dif` und desto stärker ist die Weichzeichnung.

Nun wird der gerade geänderte Weichzeichnungsfilter noch auf den Movieclip content in Zeile 8 angewandt.

Wenn Sie den Flash-Film jetzt testen, wird der Inhalt beim Scrollen verwischt.

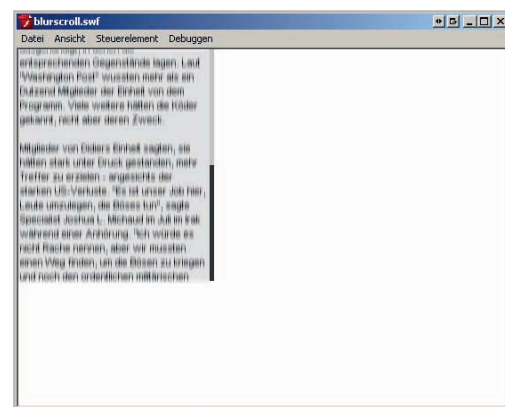


Abbildung 13 ►

Beim Scrollen verwischt der Inhalt des Movieclips.

Wellen simulieren

Kreativer Einsatz von Masken

Im Folgenden arbeiten Sie mit einer optischen Täuschung. Stellen Sie sich vor, Sie hätten ein Bild. Dieses liegt auf dem Boden eines Aquariums, in das man einen Stein wirft. Sie sehen nun eine sich kreisförmig ausbreitende Verzerrung des Bildes.

In diesem Workshop bauen Sie genau diesen Effekt mit Flash nach und verwenden den Mauszeiger als »Stein«. Eine reine Zeitleistenanimation reicht hier als Werkzeug nicht ganz aus – Sie werden ActionScript einsetzen, um ans Ziel zu gelangen.

Zielsetzungen:

Maske erstellen

Maske animieren

Maske vervielfältigen und dem Bild zuweisen

[Ergebnis: 05_WELLEN-
SIMULIEREN • WELLEN_
WS.SWF]

```
var bottomPicture:MovieClip = this.attachMovie("bild",  
"topPicture", this.getNextHighestDepth());  
bottomPicture._x = 0;  
bottomPicture._y = 0;  
  
var topPicture:MovieClip = this.attachMovie("bild",  
"bottomPicture", this.getNextHighestDepth());  
topPicture._x = 2;  
topPicture._y = 1;  
  
var mask:MovieClip = this.createEmptyMovieClip("mask",  
this.getNextHighestDepth());
```



Foto: © Fotolia, Christian Althaus

Schritt 1 Vorbereitungen

Im ersten Schritt legen Sie einen neuen Ordner auf Ihrer Festplatte an und nennen diesen WELLEN-EFFEKT. Hier werden Sie die von Ihnen erstellten Beispiele zu diesem Workshop abspeichern.

Schritt 2 Filmeinstellungen vornehmen

Erstellen Sie ein neues AS2-Dokument. Importieren Sie das auf der DVD befindliche Bild SONNENUNTERGANG.JPG über DATEI • IMPORTIEREN • IN BIBLIOTHEK IMPORTIEREN in die Bibliothek. Die Bühnengröße hängt hier vom verwendeten Bild ab und beträgt in unserem Fall 338x495 Pixel. Das ist die exakte Größe des Bildes. Als BILDRATE wählen Sie 30bps.

Schritt 3 Konvertieren des Bildes in ein Symbol

Um das Bild mit ActionScript ansprechen zu können und Zugriff auf bestimmte Eigenschaften zu haben, müssen Sie es in ein Symbol konvertieren. Dazu ziehen Sie das Bild aus der Bibliothek auf die Bühne und klicken dann mit der rechten Maustaste darauf. Wählen Sie aus dem Kontextmenü IN SYMBOL KONVERTIEREN.

Abbildung 1 ►
IN SYMBOL KONVERTIEREN



Bildgröße bestimmen

Sie können diese ermitteln, indem Sie in der Bibliothek mit der rechten Maustaste auf das importierte Bild klicken und EIGENSCHAFTEN wählen. So erhalten Sie Informationen über das jeweilige Objekt, im Falle eines Bildes u.a. die Größe.

Es öffnet sich ein Fenster, in dem »Symbol 1« als Name für das neue Symbol vorgeschlagen wird. Ersetzen Sie diesen Namen durch »bild«. Als Typ wählen Sie MOVIECLIP.

Abbildung 2 ►
Als TYP wird MOVIECLIP
zugewiesen.



Anschließend können Sie das Bild wieder von der Bühne löschen, indem Sie es markieren und **Entf** drücken.

Geben Sie dem neuen Symbol den Bezeichner »bild«. Klicken Sie hierzu in der Bibliothek mit der rechten Maustaste auf das Symbol mit dem Namen »bild«, und wählen Sie im Kontextmenü VERKNÜPFUNG.

Setzen Sie je ein Häkchen bei EXPORT FÜR ACTIONSCRIPT und bei IN ERSTES BILD EXPORTIEREN. Geben Sie »bild« im Feld BEZEICHNER ein.



Nun geht es ans Skripten. Legen Sie folgendes Skript auf Bild 1 der Hauptzeitleiste, indem Sie das Bild markieren und dann **[F9]** bzw. **[Alt] + [F9]** drücken, um das Skriptfenster zu öffnen:

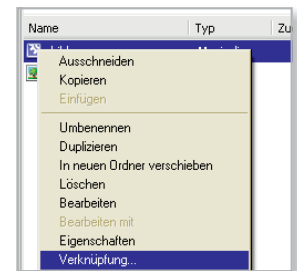
```
var bottomPicture:MovieClip = this.attachMovie("bild",  
"topPicture", this.getNextHighestDepth());  
bottomPicture._x = 0;  
bottomPicture._y = 0;  
var topPicture:MovieClip = this.attachMovie("bild",  
"bottomPicture", this.getNextHighestDepth());  
topPicture._x = 2;  
topPicture._y = 1;
```

Das Bild wird zweimal übereinander platziert, wobei das obere Bild leicht versetzt ist. Es wird mit der `attachMovie`-Methode auf die Bühne geladen. Nach dem Platzieren der Bilder werden diese jeweils mit den Eigenschaften `_x` und `_y` positioniert.

Wenn Sie den Film jetzt testen, sehen Sie die beiden Bilder im Flash-Film, wobei das eine natürlich das andere größtenteils überdeckt. Die Maske wird später Teile des versetzten Bildes »durchscheinen« lassen, wodurch der gewünschte Effekt entsteht.

Schritt 4

Platzieren des Bildes mit ActionScript



▲ **Abbildung 3**
Verknüpfungseigenschaften aufrufen

◀ **Abbildung 4**
Verknüpfungsbezeichner festlegen

Symboltypen

Ein Movieclip ist der mächtigste Symboltyp in ActionScript 2 und wird in der Regel für sämtliche Animationen verwendet. Eine Grafik verbraucht weniger Performance, verfügt aber nicht über wesentliche Eigenschaften eines Movieclips. So kann z. B. kein Instanzname vergeben werden (der jedoch nötig ist, um das Symbol mit ActionScript anzusprechen). Eine Schaltfläche verfügt über spezielle Eigenschaften, um Mausereignisse zu verarbeiten. Dies lässt sich aber auch mit Movieclips bewerkstelligen.

Schritt 5 Maske anlegen

Abbildung 5 ►
Strich- und Füll-
farbe einstellen

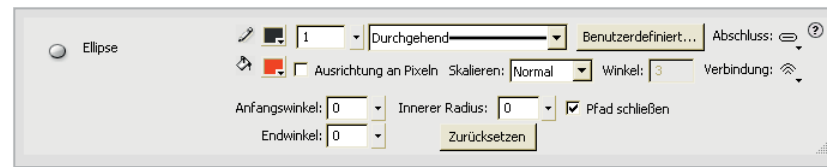


Legen Sie jetzt in der Bibliothek einen neuen Movieclip mit dem Namen »kreis« an. Klicken Sie hierzu auf das kleine Plus-Zeichen am unteren Rand des Bibliotheksfensters.

Geben Sie als Verknüpfungsname ebenfalls »kreis« ein. Stellen Sie nun Strich- und Füllfarbe ein, und wählen Sie hierfür unterschiedliche Farben. Das muss zwar nicht sein, ist jedoch übersichtlicher. Wählen Sie z. B. Schwarz als Strichfarbe und Rot als Füllfarbe.

Schritt 6 Masken-Movieclip zeichnen

Abbildung 6 ►
Stricheigenschaften festlegen

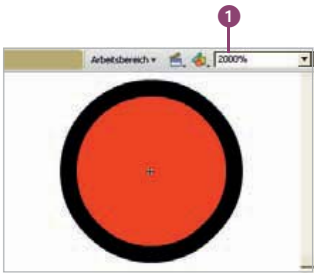


Öffnen Sie den noch leeren Movieclip durch einen Doppelklick, falls er nicht bereits geöffnet ist. Zeichnen Sie einen Kreis, dessen Mittelpunkt sich ungefähr mit dem Nullpunkt des Movieclips deckt. Dieser Kreis stellt eine einzelne Welle dar.

Den Nullpunkt des Movieclips erkennen Sie an einem kleinen Kreuz. Den Mittelpunkt der Kreisform können Sie sehen, wenn Sie das Frei-Transformieren-Werkzeug  wählen und dann die Form markieren. Wenn sich die beiden Punkte jetzt nicht decken, können Sie den Kreis ganz einfach entsprechend verschieben – Augenmaß reicht hier völlig. Da wir einen sehr kleinen Kreis benötigen, empfiehlt es sich, einen hohen Zoomfaktor einzustellen. Wenn Sie anschließend wieder den Zoomfaktor 100% einstellen, sollte der Kreis nicht größer sein als das Kreuz, das den Nullpunkt des Movieclips kennzeichnet. Bei einem Zoomfaktor von 2000% ¹ sollte die Bühne etwa wie in Abbildung 7 aussehen.

Kreis zeichnen

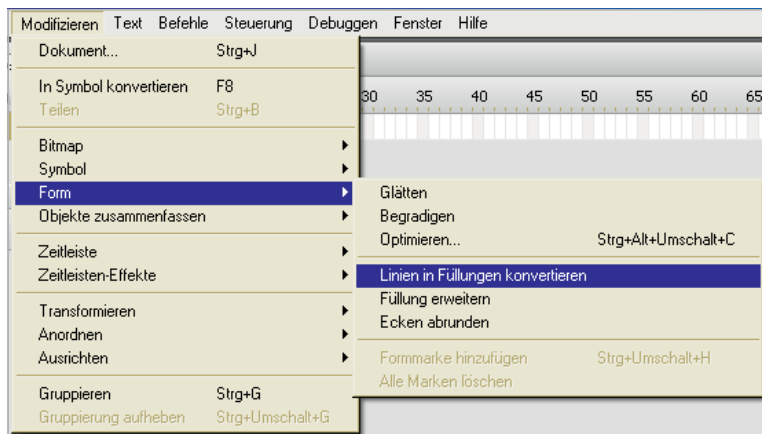
Um einen sauberen Kreis zu erhalten, halten Sie  gedrückt, während Sie den Kreis aufziehen.



▲ Abbildung 7
Der fertige Kreis

Schritt 7 Füllung entfernen und in Form konvertieren

Klicken Sie nun – am besten mit hohem Zoomfaktor – auf die rote Füllung des Kreises, und drücken Sie **Entf**. Die Kreis-Maske soll keine Füllung haben. Allerdings akzeptiert Flash nur Füllungen und Formen als Masken. Striche und Konturen werden nicht dargestellt. Sie müssen also die Kreiskontur in eine Füllung verwandeln. Markieren Sie hierzu den Kreis, und wählen Sie den Menüpunkt **MODIFIZIEREN • FORM • LINIEN IN FÜLLUNGEN KONVERTIEREN**.

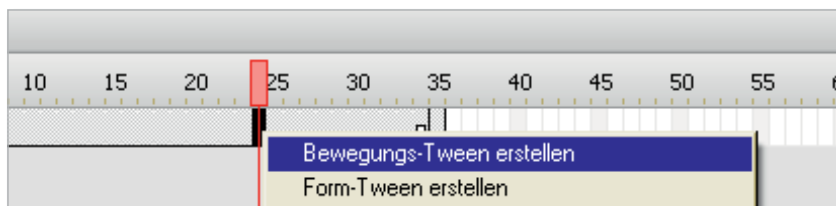


◀ **Abbildung 8**
LINIEN IN FÜLLUNGEN
KONVERTIEREN

Erst jetzt kann die Kreisform als Maske eingesetzt werden.

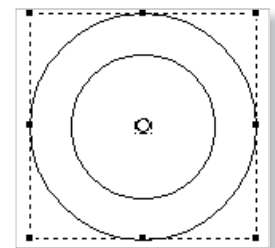
Nun benötigen Sie einen Tween, der den Kreis vergrößert. Hierzu müssen Sie die Kreisform erst einmal in einen Movieclip verwandeln. Klicken Sie mit der rechten Maustaste auf den Kreis und wählen Sie den Menüpunkt MODIFIZIEREN • IN SYMBOL KONVERTIEREN. Geben Sie als NAME »kreis_innen« und als TYP die Option MOVIECLIP an. Fügen Sie nun in Bild 30 der Zeitleiste des Movieclips »kreis« ein Schlüsselbild ein. Vergrößern Sie den Kreis proportional (mit gedrückter $\square$ -Taste) auf etwa den fünfzehnfachen Radius.

Erstellen Sie nun einen Bewegungs-Tween zwischen den beiden Schlüsselbildern, indem Sie im Kontextmenü eines beliebigen Bildes zwischen den beiden Schlüsselbildern BEWEGUNGS-TWEEN ERSTELLEN wählen.



Schritt 8

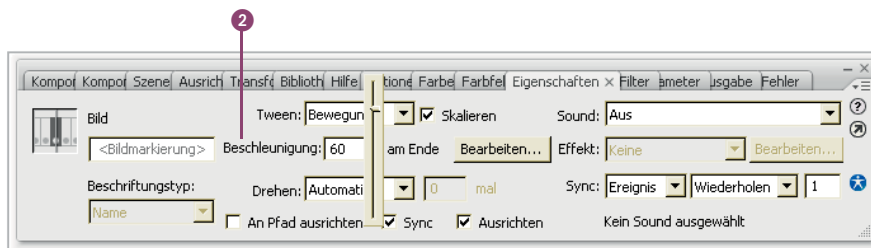
Masken-Movieclip animieren



▲ **Abbildung 9**
Skalieren des Kreises

◀ **Abbildung 10**
Bewegungs-Tween erstellen

Stellen Sie im Eigenschaftenfenster als BESCHLEUNIGUNG ② »60« ein.



◀ **Abbildung 11**
Tween-Beschleunigung
einstellen

Schritt 9

Masken-Movieclips am Ende der Animation entfernen

Wie Sie sich schon denken können, stellt dieser animierte Kreis einen einzelnen Kreis innerhalb der Maske dar. Er muss also an geeigneter Stelle vervielfacht werden. Damit sich die Kreise jedoch nicht anhäufen, sollten sie am Ende der Animation auch wieder entfernt werden. Fügen Sie hierzu eine Codezeile in Bild 30 der Zeitleiste des »kreis«-Movieclips ein:

```
removeMovieClip(this);
```

Durch diese Anweisung löscht jede Instanz des Movieclips sich selbst, sobald das Ende der Animation erreicht ist.

Schritt 10

Leere Maske auf die Bühne holen

Wechseln Sie nun wieder in die Hauptzeitleiste, und markieren Sie Bild 1, das unser bisheriges Skript enthält. Durch Drücken von **[F9]** bzw. **[Alt] + [F9]** öffnen Sie das Skriptfenster. Fügen Sie ganz unten folgende Codezeile hinzu:

```
var mask:MovieClip = this.createEmptyMovieClip("mask", this.  
    getNextHighestDepth());
```

Die Funktion `createEmptyMovieClip()` fügt einem Objekt einen neuen Movieclip ohne Inhalt hinzu, der hier als Container für die einzelnen kreisförmigen Wellen dient. Die beiden Parameter geben den Instanznamen und die Tiefe an.

Schritt 11

Maske zuweisen

In diesem Schritt definieren Sie mit ActionScript diesen neuen Movieclip als Maske für das obere Bild. Ergänzen Sie dazu am Ende des Skripts folgende Codezeile:

```
topPicture.setMask(this.mask);
```

Diese Movieclip-Funktion maskiert einen Movieclip und erwartet als Parameter einen weiteren Movieclip, der als Maske dienen soll.

In unserem Fall benutzen wir hierfür den Handler `mask`, den wir bei `createEmptyMovieClip()` festgelegt haben und der eine Referenz auf den Movieclip »mask« darstellt.

Das Bild verfügt nun über eine Maske; jedoch ist diese noch ohne Inhalt.

Im letzten Schritt müssen Sie der Maske in regelmäßigen Intervallen Instanzen der »kreis«-Movieclips hinzufügen. Hierzu definieren Sie (wieder ganz am Ende des Skripts) zunächst ein Intervall:

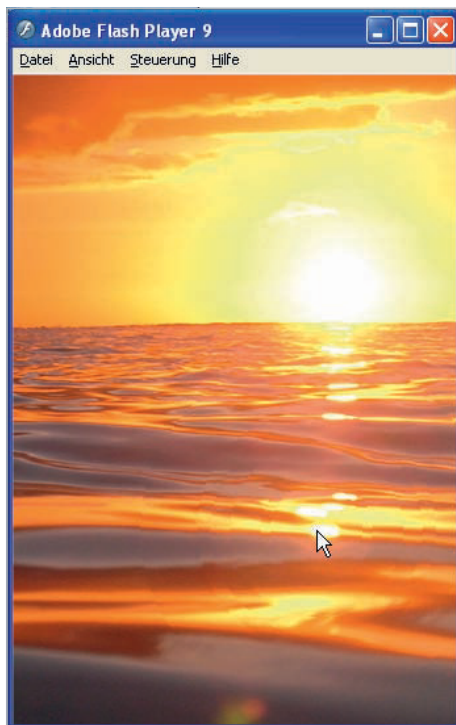
```
var intervallID:Number = setInterval(this, "waves", 200);
```

Die Funktion `setInterval()` führt hier alle 200 Millisekunden eine Funktion `waves` aus. Diese müssen Sie natürlich noch definieren. Fügen Sie folgende Codezeilen am Ende Ihres Skripts ein:

```
1: function waves() {  
2:     var circle:MovieClip = mask.attachMovie("kreis", "kreis",  
        mask.getNextHighestDepth());  
3:     circle._x = _xmouse;  
4:     circle._y = _ymouse;  
5: }
```

Zunächst wird dem Movieclip »mask« per `attachMovie()` eine Instanz des Movieclip-Symbols `kreis` hinzugefügt (Zeile 2). Anschließend wird dieser als Position die aktuelle Mausposition zugewiesen (Zeilen 3 und 4).

Testen Sie den Film – der Welleneffekt ist umgesetzt!



Schritt 12

Masken-Container mit Masken-Movieclips füllen

◀ **Abbildung 12**
Ausgehend von der Maus entstehen nun kreisförmige Wellen.

Pflanzengenerator

Pflanzen mit der Maus wachsen lassen

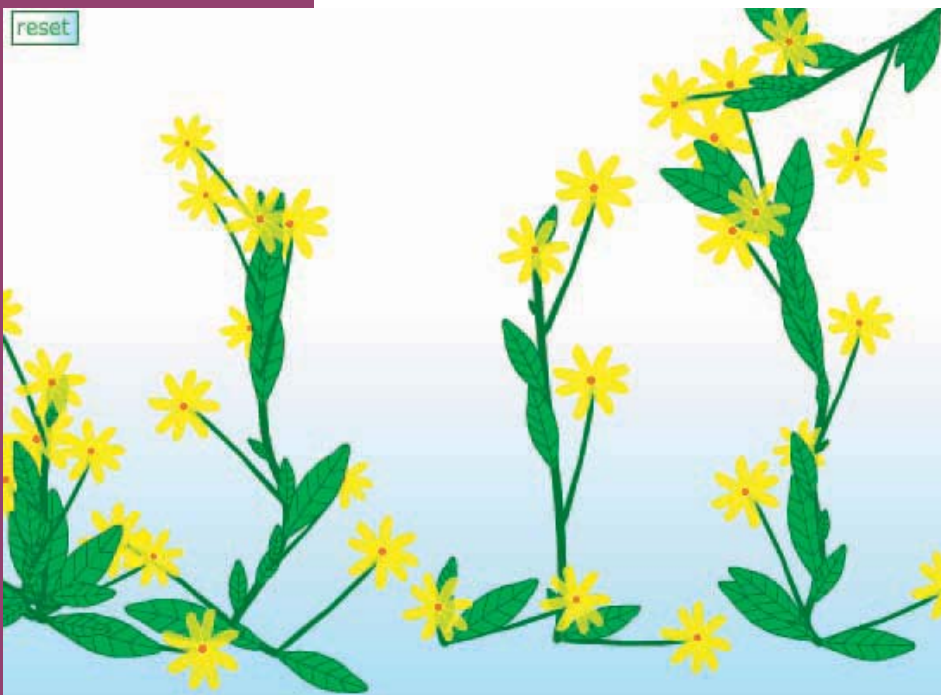
```
import flash.geom.Point;
var delay:Number;
var point:Point;
var plantContainer:MovieClip;
var plantCounter:Number;
var knobCounter:Number;
var leafCounter:Number;
var knobTimer:Number;
var leafTimer:Number;
init();
function init():Void{
    delay = 20;
    point = new Point(_xmouse,_ymouse);
    plantContainer = this.createEmptyMovieClip("container",
        this.getNextHighestDepth());
    plantCounter=0;
    knobCounter=0;
    knobTimer = getTimer();
    leafTimer = getTimer();
}
```

In diesem Workshop erstellen Sie einen hübschen Effekt, bei dem Pflanzen entlang der Mausbewegung »wachsen«. Hierbei verwenden Sie die Zeichnen-Funktionen von Flash und lernen einige Tricks kennen, die die Zeichnung weicher und organischer aussehen lassen.

Zielsetzungen:

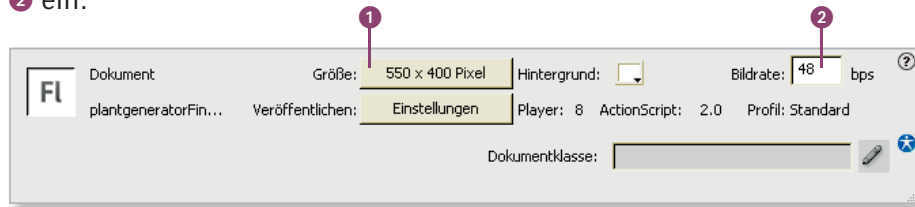
- Zeichnen mit Flash
- Zeichnungen in einzelnen Movieclips speichern
- Natürliche Bewegungen simulieren

[Ergebnis: 06_PFLANZEN-
GENERATOR • PLANT-GENERA-
TORFINAL.SWF]



Im ersten Schritt legen Sie einen neuen Ordner auf Ihrer Festplatte an und nennen ihn »Pflanzengenerator«. Hier werden Sie die von Ihnen erstellten Beispiele zu diesem Workshop abspeichern.

Erstellen Sie über DATEI • NEU eine neue Flash-Datei (ActionScript 2). Im Eigenschaftsfenster können Sie als Bühnengröße **1** die Standard-Voreinstellung von 550x400 Pixel übernehmen. Tragen Sie 48bps als BILDRA-
2 ein.



Das Zeichnen mit ActionScript ist relativ einfach. ActionScript stellt Befehle zum Zeichnen von Formen, Linien und Kurven bereit. Beim Erstellen eines eigenen kleinen Zeichenprogramms muss also nur noch die Mausbewegung erfasst und als sogenannter Parameter an die Zeichnen-Funktionen übergeben werden.

Fügen Sie in Bild 1 der Hauptzeitleiste folgendes Skript ein:

```
1: init();
2: function init():Void{
3:     this.lineStyle(5, 0x006600, 100);
4: }
5: function onMouseDown(){
6:     this.moveTo(this._xmouse, this._ymouse);
7:     this.onEnterFrame=engine;
8: }
9: function onMouseUp(){
10:    delete this.onEnterFrame;
11: }
12: function engine(){
13:    this.lineTo(this._xmouse, this._ymouse);
14: }
```

In Zeile 1 wird eine Funktion `init()` aufgerufen, die in den Zeilen 2 bis 4 definiert wird. Diese Funktion legt lediglich einen Linienstil fest, und zwar: fünf Pixel breit, dunkelgrün und nicht transparent (Zeile 3).

Schritt 1

Vorbereitungen

Schritt 2

Anlegen der Flash-Datei

◀ Abbildung 1

Das Eigenschaftsfenster erreichen Sie über FENSTER • EIGENSCHAFTEN • EIGENSCHAFTEN.

Schritt 3

Zeichnen mit ActionScript

Fertige Flash-Datei

In diesem Workshop gibt es keine Klasse. Zwar ist der Quelltext nicht trivial, aber es handelt sich um einen speziellen Effekt, dessen »Einzelteile« sich kaum für wieder verwertbare Klassen eignen. Stattdessen macht es eher Sinn, den Effekt als Ganzes in ein Projekt einzubinden. Um sich ein Bild zu verschaffen, was Sie in diesem Workshop erwartet, sehen Sie sich ruhig einmal die fertige Datei auf der DVD an, bevor Sie sich Schritt für Schritt diesem Ziel annähern. Sie finden sie im Ordner 06_PFLANZENGENERATOR.

Die Funktion `onMouseDown()` legt fest, dass immer bei Erreichen eines neuen Bildes die Funktion `engine()` ausgeführt wird. Da wir zu Beginn eine Bildrate von 48 bps festgelegt haben, wird sie 48 Mal pro Sekunde ausgeführt. Durch die Funktion `onMouseUp()` wird dies wieder beendet.

Es wird also die Funktion `engine()` immer ausgeführt, solange der Benutzer die Maustaste gedrückt hält. Die Funktion selbst zeichnet eine Linie zur jeweils aktuellen Mausposition (Zeile 13).

Testen Sie das Skript über `Strg`/`⌘` + `↵`.

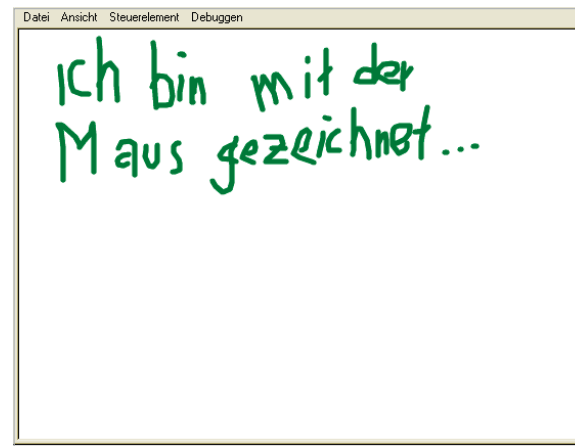


Abbildung 2 ►

Gratulation, Sie haben soeben ein Zeichenprogramm geschrieben!

Schritt 4 **Zeichnen einzelner** **Objekte**

Mit diesem nicht gerade komplexen Zeichenprogramm können Sie nur direkt auf die Bühne zeichnen. Im gesamten Programm kommt kein einziger Movieclip vor (außer der Bühne selbst, die in AS2 auch vom Typ `MovieClip` ist).

Um die gezeichneten Objekte jedoch irgendwie weiterverarbeiten zu können, müssen diese auch als Objekte im technischen Sinne angelegt werden, in diesem Fall als Movieclips. Eine anschauliche Struktur wäre zum Beispiel, einen zunächst leeren Container-Movieclip anzulegen, in den alles gezeichnet wird. Dieser Container enthält dann weitere Container-Movieclips – einen für jedes gezeichnete Objekt.

In unserem Beispiel würde eigentlich auch ein einziger Container-Movieclip genügen, da er ohnehin nur zum Löschen der Pflanzen verwendet wird. Dabei wird ganz einfach der gesamte Container mit allem Inhalt gelöscht. Mit einer Lösch-Funktion können Sie Ihre gezeichneten Pflanzen wieder entfernen, ohne jedes Mal das Programm neu starten zu müssen. Dennoch empfiehlt es sich, auch die Pflanzen in einzelne Movieclips zu packen. So halten Sie sich Erweiterungen jedweder Art offen, und das ohne großen Mehraufwand.

Der Container muss nur ein einziges Mal angelegt werden. Der entsprechende Code wird daher in der `init`-Funktion unterhalb der `LineStyle`-Anweisung ergänzt:

```
plantContainer = this.createEmptyMovieClip("container",
    this.getNextHighestDepth());
```

Außerdem werden die wichtigsten Variablen am Anfang des Skripts deklariert. Um nicht immer wieder an die entsprechende Stelle springen zu müssen, sollen hier gleich alle Variablen deklariert werden, auch wenn diese erst später im Skript zum Einsatz kommen. Ergänzen Sie also folgenden Code direkt über dem Aufruf der Funktion `init()`:

```
var delay:Number;
var point:Point;
var plantContainer:MovieClip;
var plantCounter:Number;
var knobCounter:Number;
var leafCounter:Number;
var knobTimer:Number;
var leafTimer:Number;
```

Schließlich muss noch für jedes gezeichnete Objekt – also jede Pflanze – ein eigener Movieclip erstellt werden. Dies geschieht immer dann, wenn der Benutzer die linke Maustaste drückt, um mit dem Zeichnen einer neuen Pflanze zu beginnen. Modifizieren Sie die Funktion `onMouseDown()` folgendermaßen:

```
1: function onMouseDown():Void{
2:     plantCounter++;
3:     point.x=_xmouse;
4:     point.y=_ymouse;
5:     this["line"+plantCounter] = plantContainer.
        createEmptyMovieClip("line"+plantCounter,plantContainer.
        getNextHighestDepth());
6:     this["line"+plantCounter].lineStyle(5,0x006600,100);
7:     this["line"+plantCounter].moveTo(_xmouse,_ymouse);
8:     this.onEnterFrame=engine;
9: }
```

Schritt 5

Anlegen eines Container-Movieclips

Tipp

Auch wenn Sie noch keine Erweiterung Ihres Projektes planen – achten Sie immer auf eine übersichtliche Datenstruktur. Wenn Sie mit Movieclips arbeiten, ist es oft ratsam, für jedes Element einen eigenen Movieclip zu verwenden. Direktes Zeichnen auf der Bühne ist nur selten von Vorteil.

Schritt 6

Anlegen einzelner Movieclips pro Objekt

In den Zeilen 5 bis 7 werden neue Movieclips erzeugt, mit einem Linienstil (`lineStyle`) versehen und an der aktuellen Mausposition platziert. Hierbei werden die Namen mit Hilfe der Variablen `plantCounter` dynamisch erzeugt.

Ebenso kommt eine Variable `point` ins Spiel. Diese sorgt letztendlich für ein weiches Zeichnen.

Schritt 7 Organisches Zeichnen

Damit Sie sich ein Bild davon machen können, bedarf es aber noch einiger Anpassungen. Zunächst muss die `Point`-Klasse importiert werden. Ergänzen Sie also an oberster Stelle im Skript folgende Import-Anweisung:

```
import flash.geom.Point;
```

Passen Sie dann die Funktion `init()` wie folgt an:

```
1: function init():Void{
2:     delay = 20;
3:     point = new Point(_xmouse,_ymouse);
4:     plantContainer = this.createEmptyMovieClip("container",
        this.getNextHighestDepth());
5:     plantCounter=0;
6:     knobCounter=0;
7: }
```

Wie Sie sehen, wird in Zeile 3 ein neues `Point`-Objekt erzeugt, dem die aktuelle Mausposition zugewiesen wird.

Unter einem `Point`-Objekt können Sie sich einen abstrakten Punkt vorstellen, der zwar nicht zu sehen ist, aber rechnerisch verwendet werden kann.

Außerdem wird die Variable `delay` auf »20« gesetzt. Was dies nun alles zu bedeuten hat, sehen Sie nach einer weiteren kleinen Anpassung im Skript. Ändern Sie die Funktion `engine()` folgendermaßen:

```
1: function engine():Void{
2:     point.x += (_xmouse-point.x)/delay;
3:     point.y += (_ymouse-point.y)/delay;
4:     this["line"+plantCounter].lineTo(point.x,point.y);
5: }
```

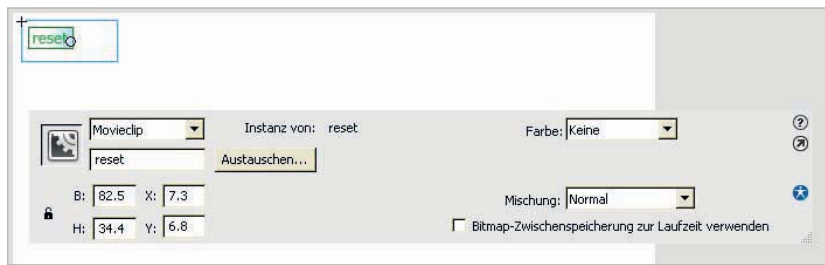
Die `lineTo`-Anweisung in Zeile 4 benutzt jetzt nicht mehr direkt die Mauskoordinaten, sondern die Werte des Point-Objekts. Diese werden zur Laufzeit immer wieder dynamisch gesetzt (Zeilen 2 und 3). Dabei wird eine Formel verwendet, die für das besagte weichere Zeichnen sorgt.

Das war jetzt eine ganze Menge Theorie, und Sie haben es sich verdient, das bisherige Ergebnis anzusehen. Testen Sie den Film über `[Strg]/[⌘]+[↵]`. Die Linie folgt dem Mauszeiger nun mit einer leichten Verzögerung, was viel organischer wirkt.

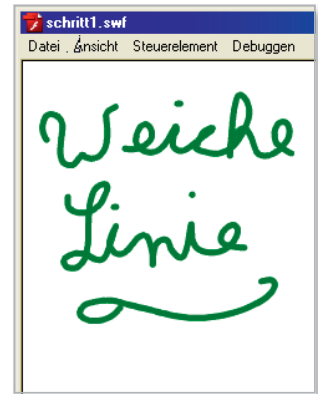
Ein Nebeneffekt dieser Zeichnen-Methode ist, dass Sie automatisch glattere Linien zeichnen. Während Sie bei einer direkten Verfolgung des Mauszeigers auch kleinste Kurven zeichnen können (und meist unfreiwillig tun, selbst wenn Sie das gar nicht wollen), wird dies nun unterbunden. Schließlich nähert sich die Linienspitze nur langsam der Mausposition, weswegen sehr enge Kurvenbewegungen mit der Maus kaum einen Effekt haben.

Da Sie nun über einen Container-Movieclip verfügen, ist es ein Leichtes, die Pflanzen wieder zu löschen. Sie müssen hierzu lediglich den Container entfernen. Erstellen Sie einen beliebigen Movieclip, der als Reset-Button dienen wird. Geben Sie diesem im Eigenschaftenfenster den Instanznamen »reset«, und ergänzen Sie ganz unten in Ihrem Skript folgende Funktion:

```
1: reset.onRelease=function(){
2:     removeMovieClip(this._parent.plantContainer);
3:     init();
4: }
```



Nun können Sie durch einen Klick auf diesen Button den gesamten Inhalt löschen und wieder von vorn beginnen. Falls nicht schon geschehen, wird hier auch ersichtlich, warum die Startinitialisierungen in eine eigene Funktion gepackt wurden. Durch den erneuten Aufruf von `init()` in Zeile 3 können Sie ganz einfach alles wieder zurücksetzen.



▲ **Abbildung 3**

Die Linie wirkt weicher und natürlicher.

Schritt 8

Reset-Funktion

◀ **Abbildung 4**

Der Reset-Button ermöglicht das Löschen der gezeichneten Objekte.

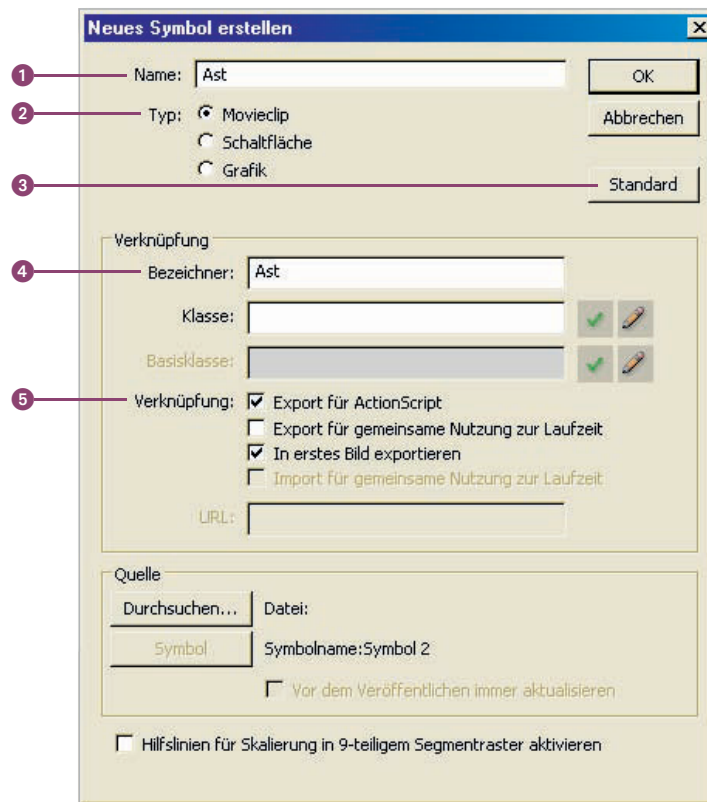
Schritt 9 Äste zeichnen

Das bisherige Ergebnis finden Sie auch auf der DVD. Die entsprechende Datei heißt SCHRITT1.FLA.

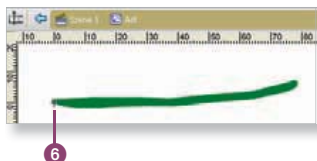
Die Kernfunktionalität ist fertig; nach Pflanzen sieht das Ganze aber noch nicht aus. Zunächst benötigen Sie Äste, an denen später Blätter und Blüten wachsen können. Das Muster für einen wachsenden Ast wird in diesem Workshop ganz einfach über einen Masken-Tween erstellt.

Erstellen Sie dazu ein neues Symbol durch einen Klick auf das Plus-Symbol in der Bibliothek. Im darauffolgenden Dialogfenster geben Sie dem Movieclip einen aussagekräftigen Symbolnamen, etwa »Ast« **1**. Wählen Sie als Typ die Option MOVIECLIP **2**, und blenden Sie über STANDARD/ERWEITERT **3** die erweiterten Einstellungsmöglichkeiten ein. Aktivieren Sie zunächst die Option EXPORT FÜR ACTIONSCRIPT **5**, und vergeben Sie anschließend den BEZEICHNER »Ast« **4**.

Abbildung 5 ▶
Ein neues Symbol wird erstellt.



▼ Abbildung 6
Ein Ast entsteht.



Zeichnen Sie in dem neuen Movieclip von links nach rechts einen grünen Strich mit einer Linienbreite von etwa 3 Pixeln und einer Länge von ca. 80 bis 100 Pixeln. Hierfür eignet sich das Stiftwerkzeug  sehr gut, da es

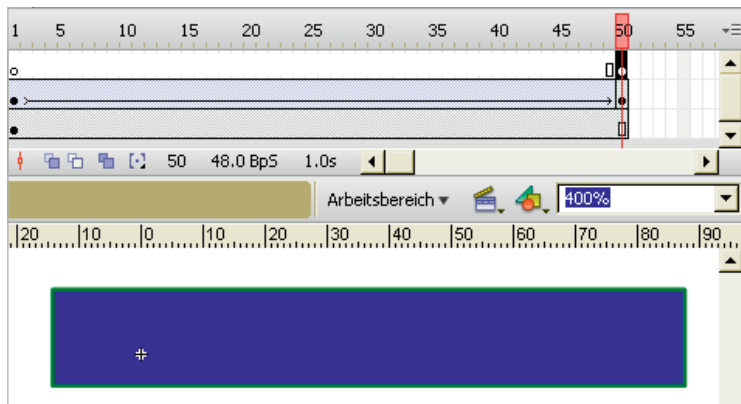
eine schöne Kurvenform ermöglicht. Außerdem ist es wichtig, dass sich der Ursprung des Astes im Mittelpunkt des Movieclips **6** befindet.

Benennen Sie die bereits vorhandene Ebene 1 um in »Ast«, und fügen Sie oberhalb der Ebene zwei weitere Ebenen mit den Namen »Maske« und »Blüte« hinzu.

Zeichnen Sie in der Ebene »Maske«, so wie Abbildung 8 gezeigt, ein Rechteck, und konvertieren Sie es über **[F8]** in ein Symbol mit beliebigem Symbolnamen. Schieben Sie das Rechteck an den linken Rand des Astes, so dass es den Anfang verdeckt.

Erstellen Sie jetzt über **[F6]** nacheinander ein Schlüsselbild in Bild 50 jeder Ebene. Skalieren Sie das Rechteck in der Ebene »Maske« in diesem Schlüsselbild auf die Breite des Astes, und verschieben Sie es anschließend so, dass es jetzt den gesamten Ast überdeckt (siehe Abbildung 9). Markieren Sie dazu das Rechteck und wählen Sie das Frei-Transformieren-Werkzeug .

Erstellen Sie danach einen Bewegungs-Tween zwischen Bild 1 und 50, indem Sie auf ein beliebiges Bild zwischen den beiden Schlüsselbildern in der Ebene »Maske« mit rechts bzw. **[Ctrl]** klicken und aus dem Kontextmenü den Eintrag BEWEGUNGS-TWEEN ERSTELLEN wählen.

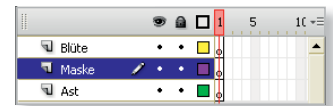


▲ **Abbildung 9**

Das Rechteck verdeckt den kompletten Ast.

Klicken Sie mit der rechten Maustaste auf die Ebene »Maske« und wählen Sie aus dem Kontextmenü den Eintrag **Maske**. Die Ebene fungiert nun als animierte Maske, die den Ast nach und nach freigibt, also »wachsen« lässt.

In diesem Schritt sollen an den Ästen Blüten wachsen. Erstellen Sie dazu wie im letzten Schritt beschrieben einen neuen Movieclip mit dem Verknüpfungsbezeichner »blume«. Zeichnen Sie, ganz nach Geschmack, eine



▲ **Abbildung 7**

Weitere Ebenen einfügen



▲ **Abbildung 8**

Das blaue Rechteck bildet die Maske und ist später im Flash-Film nicht zu sehen.



▲ **Abbildung 10**

Flash macht aus der Ebene mit dem Rechteck eine Maskenebene und rückt die Ebene »Ast« darunter ein.

Schritt 10

Blüten wachsen lassen

beliebige Blüte. Achten Sie hierbei darauf, dass der Mittelpunkt der Blüte dem Mittelpunkt des Movieclips entspricht. Die Blüte sollte, da sie später per Tween wachsen wird, nicht sehr groß sein und zunächst nur Knospengröße haben. Wählen Sie daher am besten einen sehr hohen Zoomfaktor ❶. Die einzelnen Blütenblätter können Sie beispielsweise durch Kopieren einer Musterblüte erstellen.

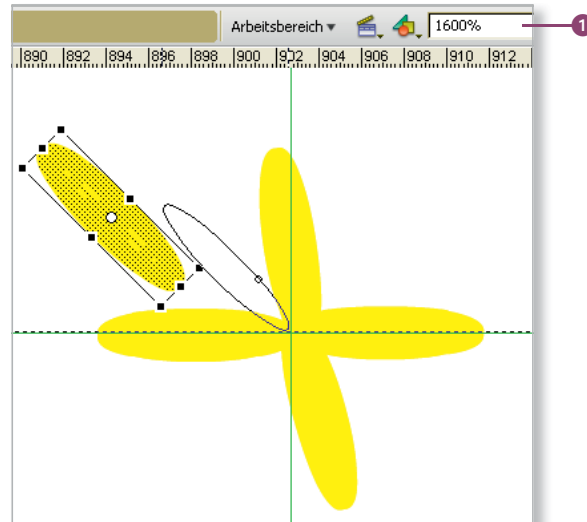


Abbildung 11 ►

Mit den Zeichenwerkzeugen von Flash entsteht Schritt für Schritt eine Blüte.

Wenn Sie mit Ihrer Blüte zufrieden sind, konvertieren Sie sie erneut in einen Movieclip. Führen Sie dazu einen Rechts- bzw. **[Ctrl]**-Klick aus, und wählen Sie den Eintrag **IN SYMBOL KONVERTIEREN**. Als **TYP** wählen Sie wieder **MOVIECLIP**. Vergeben Sie als Symbolname zum Beispiel »blute_innen«. Dieser Movieclip liegt damit im Movieclip »blume«.

Nun erstellen Sie in Bild 30 der Zeitleiste ein Schlüsselbild und skalieren dort den Blüten-Movieclip auf ca. 250%. Geben Sie ihm außerdem eine leichte Rotation sowie einen leicht verringerten Alphawert, etwa 80%.

Abbildung 12 ►

Anfangs ist die Blüte noch klein, ...

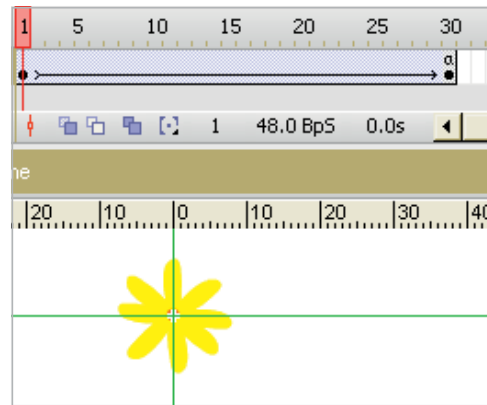
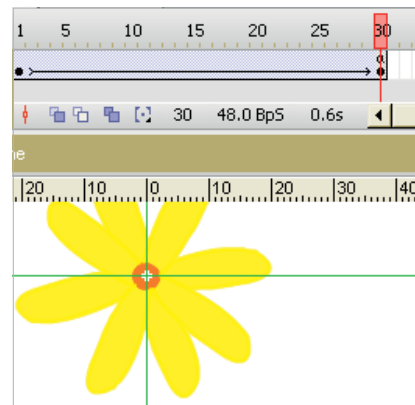
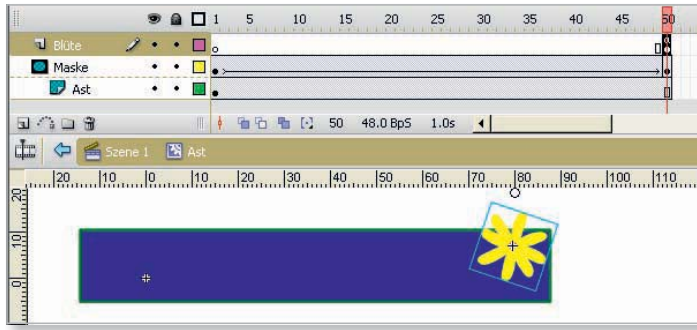


Abbildung 13 ►►

... später ist sie groß und für einen besonderen Effekt etwas transparenter.



Zu guter Letzt erstellen Sie einen Bewegungs-Tween zwischen den beiden Schlüsselbildern. Im letzten Bild fügen Sie eine `stop`-Anweisung ein. Wechseln Sie dann wieder in den Ast-Movieclip, indem Sie ihn in der Bibliothek per Doppelklick öffnen, und positionieren Sie eine Instanz der Blüte am rechten Ende des Astes in Bild 50 der Ebene »Blüte«. Der Mittelpunkt der Blüte sollte hierbei der Spitze des Astes entsprechen.



◀ **Abbildung 14**

Wenn Sie die Ebenen »Maske« und »Ast« einblenden, sollten Bühne und Zeitleiste jetzt so aussehen.

Fügen Sie in diesem Bild ebenfalls eine `stop`-Anweisung ein.

Natürlich muss Flash noch angewiesen werden, zum rechten Zeitpunkt Äste hinzuzufügen. Dafür wird eine eigene Funktion `createKnob()` definiert. Fügen Sie diese Funktion direkt unter die Funktion `engine()` ein:

```
1: function createKnob():Void{
2:     var knob:MovieClip =this["line"+plantCounter].
        attachMovie("Ast", "Ast"+knobCounter,
        this["line"+plantCounter].getNextHighestDepth());
3:     knob._x = point.x;
4:     knob._y = point.y;
5:     knob._xscale=knob._yscale=Math.random()*40+80;
6:     var rot:Number = Math.atan2(_ymouse-point.y,
        _xmouse-point.x)*180/Math.PI;
7:     var abweichung:Number = Math.random()*80-40;
8:     knob._rotation = rot+abweichung;
9:     knobCounter++;
10: }
```

In Zeile 2 wird dem aktuellen Pflanzen-Movieclip eine Instanz des Ast-Symbols hinzugefügt. In den Zeilen 3 und 4 wird diese gemäß dem Point-Objekt positioniert. Die Zeilen 6 und 7 ermitteln einen passenden Rotationswert und eine zufällige Abweichung, damit die Äste natürlich aussehen.

Schritt 11

Skriptgesteuertes Erzeugen von Ästen und Blüten

Hier können Sie ruhig mit verschiedenen Formeln und Werten experimentieren – die hier vorgeschlagene Formel erfüllt ihren Zweck aber recht gut.

Schließlich muss diese Funktion noch innerhalb der Funktion `engine()` aufgerufen werden. Ergänzen Sie oberhalb der `lineTo`-Anweisung folgenden Code:

```
if(getTimer()-knobTimer>500){
    createKnob();
    knobTimer=getTimer();
}
```

Die Variable `knobTimer` hat noch keinen Wert. Ergänzen Sie daher in der `init`-Funktion:

```
knobTimer = getTimer();
```

So wird gewährleistet, dass nur in gewissen Zeitabständen neue Äste hinzugefügt werden, was ansonsten die ganze Zeit passieren würde. Der Wert 500 ist dafür ausschlaggebend. Und auch hier lohnt es sich durchaus, mit dem Wert ein bisschen zu experimentieren.

Wenn Sie den Film jetzt testen, sehen Sie, wie die Pflanze nach und nach mit Ästen und Blüten bestückt wird. Sie finden den Zwischenstand auch auf der DVD: Es ist die Datei `SCHRITT2.FLA`.

Zwischenergebnis:
06_PFLANZENGENERATOR • SCHRITT2.FLA

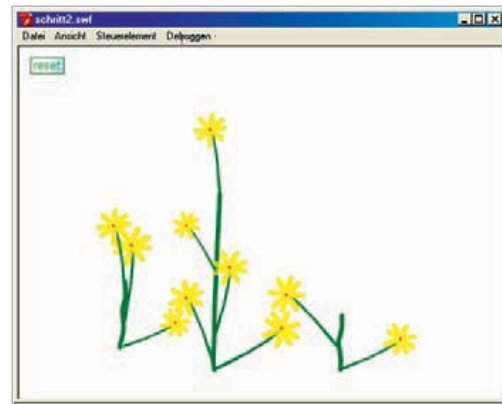


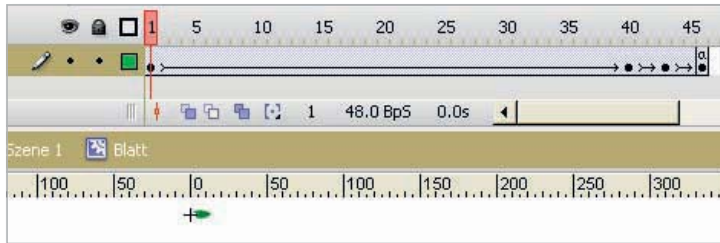
Abbildung 15 ►
Der bisherige Stand

Schritt 12

Blätter zeichnen

Aber eines fehlt noch – Blätter. Erstellen Sie wie in Schritt 9 beschrieben einen neuen Movieclip, und geben Sie diesem den Verknüpfungsbezeichner »Blatt«. In dem Movieclip zeichnen Sie mit Hilfe der Zeichenwerkzeuge von Flash ein Blatt ganz nach Ihrem Geschmack.

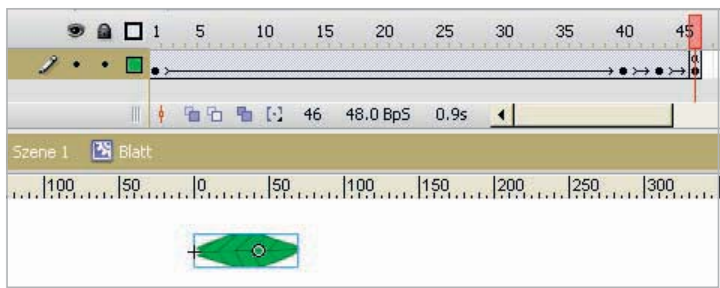
Konvertieren Sie dieses analog zu Schritt 10 in einen inneren Movieclip, und erstellen Sie ein Schlüsselbild in Bild 40. Skalieren Sie nun das Blatt in Bild 1 auf ca. 20 %, und erstellen Sie wieder einen Bewegungs-Tween zwischen den beiden Bildern. Achten Sie hierbei darauf, dass sich der Mittelpunkt des Movieclips »Blatt« immer am Ursprung des Blattes befindet – also dort, wo das Blatt den Ast berührt.



◀ **Abbildung 16**

Das Blatt ist anfangs kaum zu sehen.

Erstellen Sie nun in Bild 43 ein weiteres Schlüsselbild und eines in Bild 46. Skalieren Sie das Blatt in Bild 43 ein klein wenig, z. B. auf 95 %. So erhalten Sie einen schönen Elastik-Effekt beim Wachsenlassen der Blätter.



◀ **Abbildung 17**

Das Blatt ist fertig und auf 100 % skaliert.

Genau wie beim Zeichnen von Ästen wird es auch hier eine eigene Funktion zum Zeichnen der Blätter geben. Fügen Sie diese im Skript direkt unter der Funktion `createKnob()` hinzu:

```
function createLeaf():Void{
    var leaf:MovieClip = this["line"+plantCounter].
        attachMovie("Blatt", "Blatt"+knobCounter,
        this["line"+plantCounter].getNextHighestDepth());
    leaf._x = point.x;
    leaf._y = point.y;
    leaf._xscale=leaf._yscale=Math.random()*100;
    var rot:Number = Math.atan2(_ymouse-point.y,
    _xmouse-point.x)*180/Math.PI;
    var abweichung:Number = Math.random()*80-40;
```

Schritt 13

Blätter per Skript zeichnen

```

    leaf._rotation = rot+abweichung;
    leafCounter++;
}

```

Wie Sie sehen, arbeitet die Funktion analog zur `createKnob`-Funktion. Entsprechend muss nun auch die Funktion `engine()` wieder angepasst werden:

```

if(getTimer()-leafTimer>200){
    createLeaf();
    leafTimer=getTimer();
}

```

Und natürlich muss auch die Variable `leafTimer` einen Wert erhalten. Ergänzen Sie daher in der Funktion `init()` folgenden Code:

```

leafTimer = getTimer();

```

Schritt 14 Abschließende Blüte

Zu guter Letzt wollen wir noch eine Blüte an der Spitze jeder Pflanze wachsen lassen, die sonst etwas karg aussähe. Ergänzen Sie innerhalb der Funktion `onMouseUp()` folgenden Code:

```

bluete_oben = this["line"+plantCounter].attachMovie("blume",
"bluete", this["line"+plantCounter].getNextHighestDepth());
bluete_oben._x=point.x;
bluete_oben._y=point.y;

```

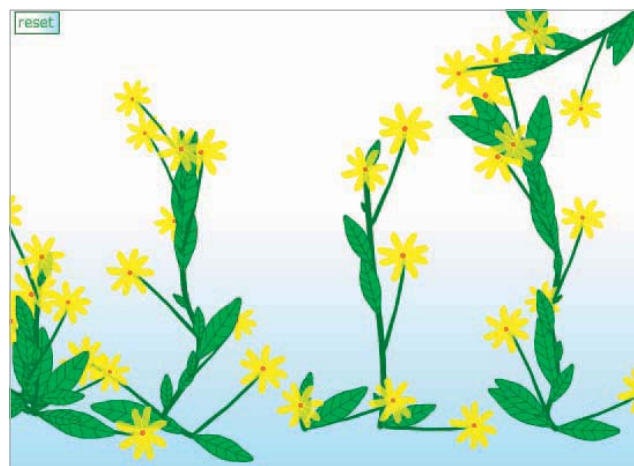


Abbildung 18 ►
Der Effekt ist fertig!

Effekte mit der Bitmap-Klasse

Das BitmapData-Objekt.....	72
TV-Rauschen	75
Verwischen-Effekt.....	79
Vergrößerungs-Effekt	87
Farbwähler	96
Anzeigeliste und attachMovie().....	104
Partikel-Effekt.....	107
Flüssigkeit simulieren.....	120
Feuer-Effekt	131
Web 2.0-Reflexion.....	141

Das BitmapData-Objekt

Es ging ein Raunen durch die Entwicklergemeinde von Flash, als mit der Veröffentlichung von Flash 8 das BitmapData-Objekt in das Flash-Plugin eingeführt wurde.

Flash rechnet mit Vektoren. Wenn man bisher 1 Pixel dynamisch zeichnen wollte, musste man eine 1 x 1 Pixel große Vektorfläche auf die Bühne zeichnen. Natürlich konnte man auch eine 1 x 1 Pixel große Bitmap importieren. Aber Flash konnte auf die Bildinformationen nicht zugreifen. Wenn man das Bild ändern oder anpassen wollte, musste man es in Photoshop oder Fireworks laden, verändern und neu abspeichern.

Seit Flash 8 ist es möglich, Pixelinformationen zur Laufzeit dynamisch zu lesen und zu schreiben. So werden Effekte möglich, die man von Photoshop kennt, nur mit dem Unterschied, dass sie in Flash animiert werden können.

Im Zuge der Einführung von Flash Player 9 und Flash CS3 hat Flash in Sachen BitmapData noch einmal einen großen Schritt nach vorne gemacht. Durch die neue Virtual Machine (AVM2) für ActionScript 3 sind die Bitmap-Operationen um ein Vielfaches schneller geworden.

Wichtig zu wissen ist noch: Das Flash-Plugin »arbeitet in Software«, was bedeutet, dass für das Rendering keine Hardwareunterstützung der Grafikkarte genutzt wird. Somit wird die Rendergeschwindigkeit von Flash – bis auf weiteres – den Programmiersprachen, die direkt auf den Prozessor der Grafikkarte zugreifen können, unterlegen sein.

Flash kann Bitmaps verstehen

In Flash war es immer möglich, Bitmaps zu importieren und anzuzeigen. Aber sie konnten nicht dynamisch verändert werden. Flash konnte sie nicht lesen oder verstehen. Das ist seit Flash 8 anders. Um Bitmapdaten für Ihr Skript verfügbar zu machen, reichen zwei Zeilen:

```
1: var meinBitmapData:BitmapData = new BitmapData(200, 200);
2: meinBitmapData.draw(meinUIObject);
```

In Zeile 1 erstellen Sie ein neues BitmapData-Objekt, das Sie in Zeile 2 mit den Bildinformationen eines beliebigen UIObject (Movieclip, Bitmap, Shape etc.) befüllen. Das wars! Flash hat in die Variable meinBitmapData ein Abbild von meinUIObject in Pixel gezeichnet. Nun können Sie die Bild-daten aus meinBitmapData auslesen und/oder setzen.

Pixel auslesen

Um den Farbwert eines Pixels auszulesen, nutzen Sie folgende Funktion:

```
3: trace(meinBitmapData.getPixel(1, 1));
```

Mit dieser Funktion geben Sie den Farbwert des ersten Pixels Ihres BitmapData-Objekts `meinBitmapData` aus.

Pixel setzen

Um die Farbe eines Pixels z. B. mit einem roten Farbwert zu setzen, nutzen Sie folgende Funktion:

```
4: meinBitmapData.setPixel(1, 1, 0xFF0000);
```

BitmapData anzeigen

Ein BitmapData-Objekt existiert nur im Speicher des Flash Players. Um die Bitmapdaten anzuzeigen, benötigen Sie einen Anzeigecontainer. In Flash 8 (und AS2) konnten Sie ausschließlich Movieclips als Container benutzen:

```
5: meinMovieclipContainer.attachBitmap(meinBitmapData, 1);
```

In Flash CS3 (und AS3) gibt es das Anzeige-Objekt `Bitmap`.

```
6: meinBitmap.bitmapData = meinBitmapData;
```

Mit dieser Zeile weisen Sie das Bitmap-Objekt `meinBitmap` an, die Daten von `meinBitmapData` anzuzeigen.

Filter auf Bitmap anwenden

In Flash 8 wurden Filter eingeführt, die (ähnlich wie in Photoshop) auf Movieclips angewendet werden können. Wollen Sie einen Filter auf ein BitmapData-Objekt anwenden, machen Sie dies folgendermaßen:

```
7: var blur:BlurFilter = new BlurFilter(4, 4, 3);  
8: meinBitmapData.applyFilter(meinBitmapData, meinBitmapData.  
    rect, new Point(0, 0), blur);
```

TIPP

Im Unterschied zur `getPixel`-Methode gibt Ihnen die `getPixel32`-Methode auch den Alphawert des ausgelesenen Pixels zurück (ARGB).

In Zeile 7 erstellen Sie ein neues `BlurFilter`-Objekt. Diesem übergeben Sie die benötigten Initialisierungsparameter. Sie entscheiden, wie der `BlurFilter` genau gezeichnet wird. Detaillierte Informationen zum `BlurFilter` (und auch zu allen anderen Filtern) entnehmen Sie bitte der Flash-Hilfe.

In der Methode `applyFilter` in Zeile 8 geben Sie im ersten Parameter an, dass `meinBitmapData` die Ausgangsbitmap ist. Außerdem soll der Filter auf die gesamte Fläche von `meinBitmapData` angewandt werden (`meinBitmapData.rect`), ab der oberen linken Ecke (`new Point(0, 0)`). Als vierten Parameter schließlich übergeben Sie den Filter.

Fazit

Sie haben jetzt einen kleinen Einblick in die Möglichkeiten der Bitmap-Operationen in Flash bekommen. Jetzt liegt es an Ihnen, damit zu spielen und zu experimentieren.

Lassen Sie sich von den `BitmapData`-Workshops inspirieren. Die Möglichkeiten sind, spätestens seit der Einführung von Flash CS3 und der stark verbesserten Geschwindigkeit von AS3, praktisch unbegrenzt.

TV-Rauschen

Das Rauschen eines Fernsehers simulieren

Um ein TV-Rauschen in Flash zu erstellen, benötigen Sie bis Flash 8 noch eine Einzelbildanimation. Mit der in Flash CS3 eingeführten BitmapData-Klasse und der noise-Methode lässt sich ein TV-Rauschen nun mit Hausmitteln bewerkstelligen. Dies reduziert die Dateigröße und macht Ihre Animation nachträglich leichter veränderbar.

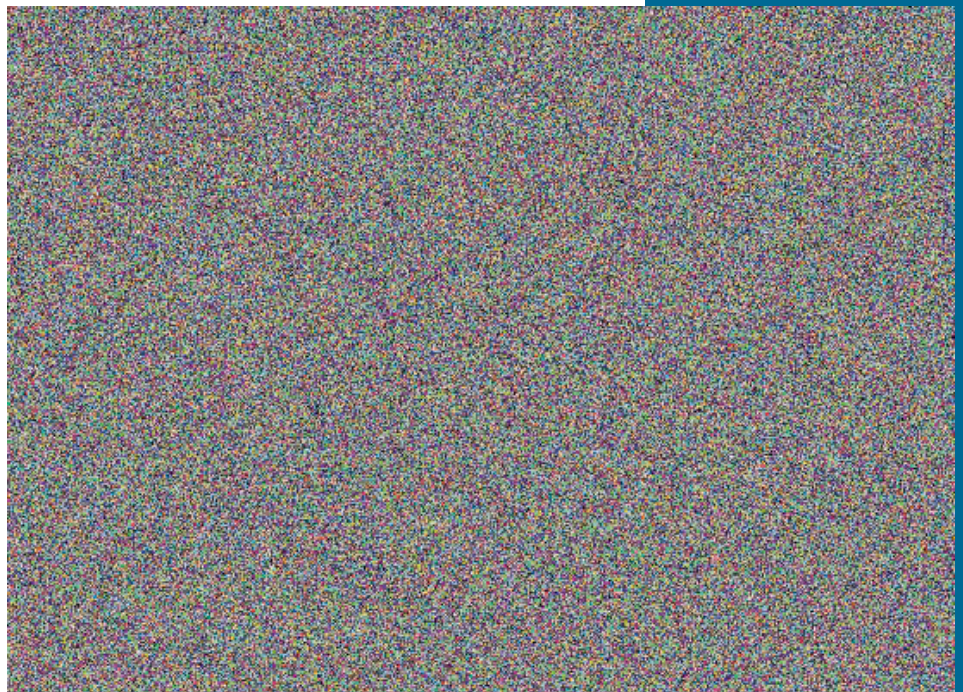
Zielsetzungen:

BitmapData-Objekt
anwenden

Mit der noise-Methode
arbeiten

[Ergebnis: 07_TV-RAUSCHEN • TVEFFEKT.SWF]

```
import flash.display.BitmapData;
rausch_mc = _root.createEmptyMovieClip("rausch_mc", 1);
rausch_bd = new BitmapData(550, 400, false, 0xFF0000);
rausch_mc.attachBitmap(rausch_bd, 1);
rausch_bd.noise(1, 0, 255, 1|2|4, false);
this.onEnterFrame = function(){
    rausch_bd.noise(Math.floor(1000*Math.random()), 0, 255,
    1|2|4, false);
}
```



Schritt 1 Dokument erstellen

Erstellen Sie ein neues AS2-Flash-Dokument, und öffnen Sie mit **Strg**/**⌘** + **J** die Dokumenteigenschaften. Stellen Sie die GRÖSSE auf 550x400 Pixel und die BILDRATE auf 25 bps ein. HINTERGRUNDFARBE und sonstige Einstellungen spielen keine Rolle.

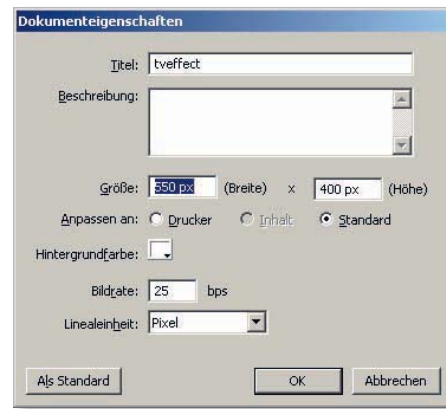


Abbildung 1 ►
Dokumenteigenschaften
festlegen

Schritt 2 Skript vorbereiten

Zunächst müssen Sie die BitmapData-Klasse für Ihr Skript verfügbar machen. Markieren Sie das erste Schlüsselbild, und rufen Sie das AKTIONEN-Fenster über FENSTER • AKTIONEN oder **F9** bzw. **Alt** + **F9** auf. Kopieren Sie folgenden Code hinein:

```
import flash.display.BitmapData;
```

Schritt 3 Ausgabe-Movieclip erstellen

Nun erstellen Sie mit ActionScript einen Movieclip, der als Container für ein BitmapData-Objekt dienen wird. Fügen Sie folgende Zeile ein:

```
rausch_mc = _root.createEmptyMovieClip("rausch_mc", 1);
```

Schritt 4 BitmapData-Objekt für das Rauschen

Jetzt benötigen Sie ein BitmapData-Objekt, das das TV-Rauschen beinhalten wird. Ergänzen Sie Ihr Skript um folgende Zeilen:

```
rausch_bd = new BitmapData(550, 400, false, 0xFF0000);  
rausch_mc.attachBitmap(rausch_bd, 1);
```

In der ersten Zeile erstellen Sie ein BitmapData-Objekt, das so groß wie die Bühne ist (550 und 400), keine Transparenzen unterstützen soll (**false**) und die Farbe Rot hat (0xFF0000).

Tipp

Weitere Informationen zur BitmapData-Klasse finden Sie im BitmapData-Exkurs auf Seite 72.

BitmapData-Objekte existieren nur im Speicher. Um sie anzuzeigen, müssen sie einem Movieclip angefügt werden. In der zweiten Zeile fügen Sie das BitmapData-Objekt `rausch_bd` an den zuvor erstellten Movieclip `rausch_mc` an (engl.: `attach`).

Wenn Sie Ihr Dokument nun mit `[Strg]/[⌘]+[↵]` oder **STEUERUNG • FILM TESTEN** veröffentlichen, sehen Sie eine rote Hintergrundfläche.

Nun werden Sie das eigentliche Rauschen erstellen. Fügen Sie in Ihrem Skript folgende Codezeile ein:

```
rausch_bd.noise(1, 0, 255, 1|2|4, false);
```

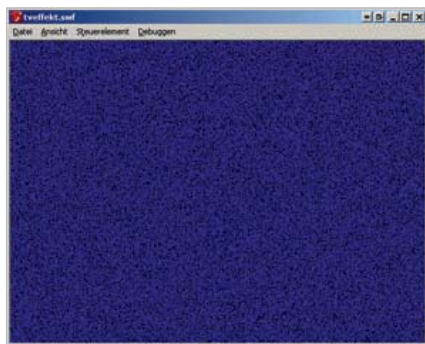
Sie übergeben der `noise`-Methode als ersten Parameter einen Zufallswert (das sogenannte `randomSeed`), in diesem Fall einfach `1`. Aus dieser Zahl berechnet Flash das Rauschen. Sie erhalten dabei für jede Zahl ein anderes Rauschen. Mehr dazu erfahren Sie im nächsten Schritt.

Im zweiten und dritten Parameter (`0` und `255`) geben Sie an, in welcher Farbspanne sich das Rauschen bewegen soll. In unserem Beispiel nutzen wir die gesamte RGB-Farbskala zwischen den Farbwerten `0` und `255`. Sie können auch mit kleineren Farbspannen experimentieren.

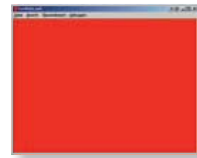
Der vierte Parameter gibt die Farben an, die für das Rauschen genutzt werden sollen: `1` (Rot), `2` (Grün) und `4` (Blau) können beliebig kombiniert werden. Wenn Sie hier also nur `4` angeben, wird das Rauschen mit der Farbe Blau in der Farbspanne zwischen `0` und `255` erzeugt. Geben Sie hier `1|4` an, wird das Rauschen aus den Farben Rot und Blau erzeugt.

Mit dem letzten Parameter schließlich können Sie (häufig gewünscht) festlegen, ob das Rauschen nur Grauwerte enthalten soll. Hier setzen wir diesen Wert jedoch auf `false`.

Exportieren Sie Ihren Flash-Film mit der Farbe Blau (`4`) über **STEUERUNG • FILM TESTEN** oder `[Strg]/[⌘]+[↵]`, und schauen Sie sich das Ergebnis an.



◀ **Abbildung 3**
Flash erzeugt ein blaues Rauschen.



▲ **Abbildung 2**
Das BitmapData-Objekt wird angezeigt.

Schritt 5

Rauschen einstellen

RGB-Farbwert

Jede Farbe setzt sich aus einer Mischung der Farben Rot (R), Gelb (G) und Blau (B) zusammen. Die Farben bewegen sich in einer Spanne von `0` bis `255`. Die Farbe Rot wird als Hexadezimalzahl `FF0000` beschrieben. Die ersten zwei Stellen geben den Rotwert an (`FF`, entspricht `255` als Dezimalzahl), die zweiten beiden Stellen den Gelbwert (`00`, entspricht `0` als Dezimalzahl) und die letzten beiden Stellen den Blauwert (`00`, entspricht `0` als Dezimalzahl).

Der Farbwert `BBBBBB` ist also eine Farbe, die sich zu jeweils gleichen Teilen (`BB = 187`) aus allen drei Farben zusammensetzt. Es handelt sich somit um ein dunkles Grau.

Mehr Informationen zum Dezimal-/Hexadezimalsystem finden Sie z. B. hier:

- ▶ <http://de.wikipedia.org/wiki/Dezimalsystem>
- ▶ <http://de.wikipedia.org/wiki/Hexadezimalsystem>

Schritt 6

Rauschen animieren

Bisher sehen Sie zwar ein Rauschen, es wird aber nur statisch angezeigt. Das Rauschen muss also animiert werden.

Fügen Sie Ihrem Skript folgenden Code hinzu:

```
this.onEnterFrame = function(){
    rausch_bd.noise(Math.floor(1000*Math.random()), 0, 255,
    1|2|4, false);
}
```

Die `onEnterFrame`-Funktion bewirkt, dass nachfolgender Code in jedem Frame, also 25 Mal pro Sekunde, ausgeführt wird.

In der nächsten Zeile finden Sie die schon bekannte `noise`-Methode. Hier ändert sich der Wert von `randomSeed` jedoch in jedem Durchlauf.

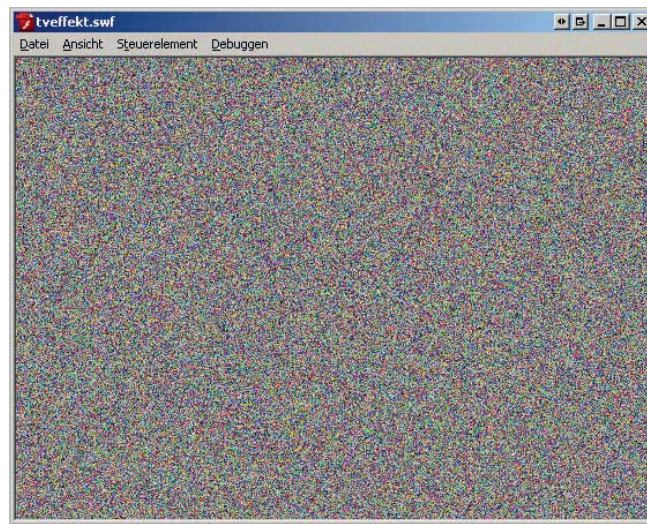
Die Methode `Math.random()` erzeugt eine Zufallszahl zwischen 0 und 1. Diese multiplizieren Sie mit 1000 und runden sie mit `Math.floor()` auf eine Ganzzahl (also eine Zahl ohne Nachkommastellen) ab.

Zufallszahlen

`Math.random()` gibt eine Pseudozufallszahl zwischen 0 und 1 zurück. `random(n)` gibt (nur in AS2) eine Zufallszahl zwischen 0 und n zurück.

Abbildung 4 ►

Das Rauschen wurde animiert und zeigt den gewünschten Effekt.



Verwischen-Effekt

Bitmapdaten abhängig von der Mausposition kopieren und verwischen

In diesem Workshop lernen Sie, wie Sie einen Verschmieren-Effekt – ähnlich dem Effekt des Photoshop-Tools Wischfinger – erstellen. Um den Eindruck des Verwischens, der Unschärfe, zu erwecken, werden wir kreisförmige Bildteile so häufig ineinander kopieren, dass die kopierten Bildteile verschwimmen und so der gewünschte Eindruck entsteht.

Zielsetzungen:

Bildteile kopieren
und einfügen
Maussteuerung erstellen
Eindruck des Verwischens
erzeugen

[Ergebnis: 08_VERWISCHEN-
EFFEKT • SMUDGE.SWF]

```
var original_bd = flash.display.BitmapData.  
loadBitmap("original");  
var output_bd = new flash.display.BitmapData(original_bd.width,  
original_bd.height, true, 0);  
output_bd.draw(original_bd);  
this.createEmptyMovieClip("image_mc", 1);  
image_mc.attachBitmap(output_bd, 1);  
var brush_bd = new flash.display.BitmapData(40, 40, true, 0);  
var blur = new flash.geom.ColorTransform();  
blur.alphaMultiplier = .2;  
brush_bd.draw(circle_mc, new flash.geom.Matrix(), blur);  
var brushed_part_bd = new flash.display.BitmapData(40, 40, true,  
0);  
this.onMouseDown = function(){  
    this.onMouseMove = onBrushMove;  
}  
this.onMouseUp = function(){  
    delete this.onMouseMove;  
}  
function onBrushMove(){
```



Schritt 1 Dokument erstellen

Erstellen Sie ein neues AS2-Flash-Dokument in der Größe 550 x 400 Pixel und mit einer BILDRATE von 25 bps. HINTERGRUNDFARBE und sonstige Einstellungen spielen keine Rolle.



Abbildung 1 ►
Legen Sie die Dokument-
eigenschaften fest.

Schritt 2 Bild importieren

Importieren Sie über DATEI • IMPORTIEREN • IN BIBLIOTHEK IMPORTIEREN das Bild ROM.JPG. Es hat exakt die Größe der Bühne. Rufen Sie nun über FENSTER • BIBLIOTHEK oder `Strg`/`⌘` + `L` die Bibliothek auf.

Nun müssen Sie dem importierten Bild einen eindeutigen Verknüpfungsnamen geben. Dazu klicken Sie mit der rechten Maustaste auf das Bild in der Bibliothek, wählen VERKNÜPFUNG, setzen in dem sich nun öffnenden Popup-Fenster das Häkchen bei EXPORT FÜR ACTIONSCRIPT und geben unter BEZEICHNER »original« ein.



Abbildung 2 ►
Weisen Sie dem importierten
Bild den Bezeichner zu.

Schritt 3 Kreis erstellen

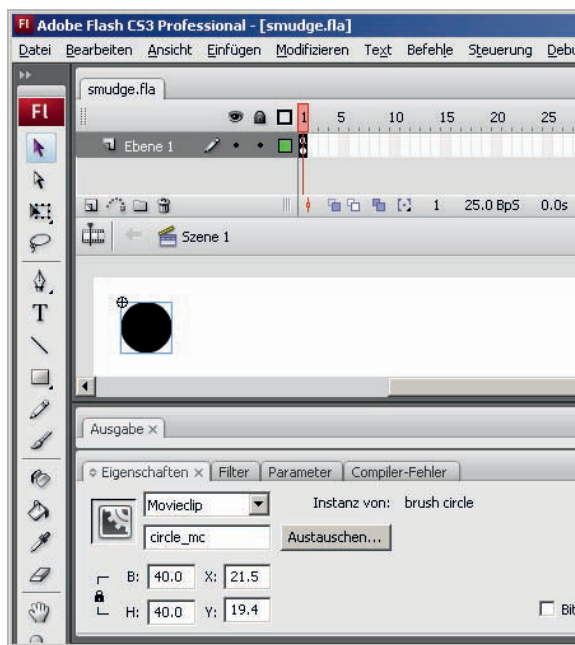
Erstellen Sie nun auf der Hauptzeitleiste mit dem Ellipsenwerkzeug  einen schwarzen Kreis mit einer Kantenlänge von 40x40 Pixel. Diesen Kreis wandeln Sie mit MODIFIZIEREN • IN SYMBOL KONVERTIEREN... oder `F8` in einen Movieclip um und geben ihm in dem sich nun öffnenden Fenster den Symbolnamen »brush circle«.



◀ **Abbildung 3**

Konvertieren Sie den gerade erstellten Kreis in einen Movieclip mit dem Namen »brush circle«.

Anschließend geben Sie dem Movieclip im Eigenschaftenfenster den Instanznamen »circle_mc«. Es spielt keine Rolle, wo der Movieclip platziert ist – er kann auch außerhalb des sichtbaren Bereichs der Bühne liegen.



◀ **Abbildung 4**

Geben Sie dem Movieclip den Namen »circle_mc«.

Jetzt sorgen Sie dafür, dass Ihr Bild aus der Bibliothek auf der Bühne angezeigt wird. Markieren Sie das Schlüsselbild in der Zeitleiste und rufen Sie das AKTIONEN-Fenster über FENSTER • AKTIONEN oder über **[F9]** bzw. **[Alt] + [F9]** auf. Kopieren Sie folgenden Code hinein:

```
1: var original_bd = flash.display.BitmapData.  
   loadBitmap("original");  
2: var output_bd = new flash.display.BitmapData  
   (original_bd.width, original_bd.height, true, 0);  
3: output_bd.draw(original_bd);
```

Schritt 4

Importiertes Bild für das Skript verfügbar machen

```

4:  this.createEmptyMovieClip("image_mc", 1);
5:  image_mc.attachBitmap(output_bd, 1);

```

In Zeile 1 laden Sie die Bitmapdaten des Bildes aus der Bibliothek (`original`) in das neue `BitmapData`-Objekt `original_bd` und machen es so für Ihr Skript verfügbar. `original_bd` enthält nun alle Pixeldaten Ihres in die Bibliothek importierten Bildes.

Dann erstellen Sie in Zeile 2 ein leeres `BitmapData`-Objekt mit dem Namen `output_bd`, das dieselben Abmessungen wie das Originalbild hat. Das Objekt dient als Container für die Bilddaten, die später ausgegeben (also dargestellt) werden.

In Zeile 3 befüllen Sie `output_bd` mit den Bitmapdaten des zuvor erstellten `original_bd`. Um die Bitmapdaten schließlich anzuzeigen, erstellen Sie in Zeile 4 einen leeren `Movieclip` und übergeben diesem die Bilddaten (Zeile 5). Wenn Sie Ihren Flash-Film mit `Strg`/`⌘` + `↵` exportieren, sehen Sie, dass das Bild aus der Bibliothek auf der Bühne angezeigt wird.



Abbildung 5 ►

Das Bild aus der Bibliothek wird auf der Bühne angezeigt.

Schritt 5 Pinsel erstellen

Jetzt erstellen Sie den Pinsel (in Photoshop Werkzeugspitze genannt), also die Fläche, mit der später die Pixel des Originalbildes verwischt werden. Fügen Sie dazu Ihrem Quellcode folgende Zeilen hinzu:

```

1:  var brush_bd = new flash.display.BitmapData(40, 40, true, 0);
2:  var blur = new flash.geom.ColorTransform();
3:  blur.alphaMultiplier = .2;

```

In Zeile 1 erstellen Sie ein leeres `BitmapData`-Objekt mit dem Namen `brush_bd`, das die Größe (40x40 Pixel) des zuvor erstellten Kreises hat.

Um diesem Objekt einen Alphawert zuweisen zu können, wird in Zeile 2 ein ColorTransform-Objekt erzeugt. Diesem wird in Zeile 3 ein `alphaMultiplier` von 0.2 zugewiesen, also ein Alphawert von 20%.

Für Ihren Pinsel benötigen Sie nun die Bitmapdaten des zuvor erstellten Kreises. Die `draw`-Methode erlaubt es Ihnen, einen Movieclip von der Bühne sozusagen »abzufotografieren«. Dabei spielt es keine Rolle, ob der Movieclip sichtbar ist – er kann auch den Alphawert 0% haben, unsichtbar oder von anderen Objekten verdeckt sein.

Fügen Sie am Ende Ihres Skripts folgende Zeile ein:

```
brush_bd.draw(circle_mc, new flash.geom.Matrix(), blur);
```

Sie übergeben der `draw`-Methode also den Movieclip, dessen Bitmapdaten Sie benötigen (`circle_mc`). Als zweiten Parameter übergeben Sie eine Standardmatrix. Um dem BitmapData-Objekt `brush_bd` einen Alphawert zuzuweisen, wird als dritter Parameter das zuvor erstellte ColorTransform-Objekt übergeben.

Wie immer existieren BitmapData-Objekte nur im Speicher des Flash Players. Um diese nun sichtbar zu machen, benötigen Sie wiederum einen Movieclip als Container. Sie erreichen dies, wenn Sie folgende Codezeilen ergänzen:

```
this.createEmptyMovieClip("test_mc", 2);  
test_mc.attachBitmap(brush_bd, 1);
```

Wenn Sie das Flash-Dokument nun mit **STEUERUNG • FILM TESTEN** oder `Strg`/`⌘` + `↵` exportieren, werden Sie – zusätzlich zu dem Bild – in der linken oberen Ecke einen Kreis mit einem Alphawert von 20% sehen.

Löschen Sie diese beiden Zeilen anschließend wieder.



Schritt 6

Pinsel anzeigen

◀ **Abbildung 6**
Der Kreis wird mit einem Alphawert von 20% oben links angezeigt.

Schritt 7

Bildteil speichern

Jetzt werden Sie den Verwischen-Effekt erstellen. Ersetzen Sie die beiden in Schritt 6 eingefügten Zeilen mit folgender Codezeile:

```
var brushed_part_bd = new flash.display.BitmapData(40, 40, true, 0);
```

Nun gehts ans Eingemachte. Sie erstellen ein weiteres BitmapData-Objekt, in dem der Bildteil gezeichnet wird, der von unserem Pinsel verwischt wird. Dieser Bildteil hat die Größe 40x40 Pixel (also die Größe unseres Pinsels) und ist zunächst leer.

Schritt 8

Mausbewegungen verfolgen

Um die Mausbewegungen des Nutzers abzufangen, erstellen Sie als Nächstes die Methoden `onMouseDown()` und `onMouseUp()`. Fügen Sie den Code ans Ende Ihres Skripts ein.

```
1: this.onMouseDown = function(){
2:     this.onMouseMove = onBrushMove;
3: }
4: this.onMouseUp = function(){
5:     delete this.onMouseMove;
6: }
```

Zeile 2 bewirkt, dass Ihr Skript, sobald der User die linke Maustaste gedrückt hält, bei jeder Mausbewegung (`onMouseMove()`) die weiter unten beschriebene Funktion `onBrushMove()` ausführt. Wenn der Nutzer die Maustaste loslässt, wird dies wieder rückgängig gemacht (Zeile 5).

Schritt 9

Kopierareal festlegen

Kommen wir nun zu unserer Kernfunktion, die Sie wieder ganz unten im Skript einfügen. `onBrushMove()` wird immer ausgeführt, wenn der Nutzer die linke Maustaste gedrückt hält und bewegt. In Zeile 2 errechnet das Skript zunächst den Punkt, auf dem sich die Maus gerade befindet. Da sich der Kreismittelpunkt direkt unter der Maus befinden soll (und nicht seine obere linke Ecke), wird von der x- und y-Position der Maus jeweils die halbe Kantenlänge des Kreises abgezogen.

```
1: function onBrushMove(){
2:     var point = new flash.geom.Point(this._xmouse - 20,
3:                                     this._ymouse - 20);
4:     var draw_rectangle = new flash.geom.Rectangle(point.x,
5:                                                     point.y, 40, 40);
```

Anschließend (Zeile 3) erstellen Sie ein Rectangle-Objekt. Dies ist sozusagen das Areal, aus dem der Bildinhalt für unseren »Wischer« kopiert wird. Dazu übergeben Sie dem Rectangle-Konstruktor zunächst die x- und y-Koordinaten des zuvor erstellten Punkts sowie die Kantenlänge unseres Pinsels (also des Kreises), nämlich 40x40 Pixel.

Um die Funktion zu vervollständigen, ergänzen Sie die folgenden Zeilen in Ihr Skript. In der ersten Zeile verändern Sie die Pixeldaten von `output_bd`, indem Sie Pixel, die in `brushed_part_bd` kopiert wurden, in `output_bd` hineinkopieren. Sie übergeben der Methode `copyPixels()` als ersten Parameter die zu kopierenden Bitmapdaten (`brushed_part_bd`), dann das Quadrat von `brushed_part_bd`. Als dritter Parameter wird der Punkt (also die Position), an dem die Pixeldaten eingefügt werden sollen, übergeben:

```
output_bd.copyPixels(brushed_part_bd, brushed_part_
    bd.rectangle, point, brush_bd, new flash.geom.Point(0, 0),
    true);
brushed_part_bd.copyPixels(output_bd, draw_rectangle, new
    flash.geom.Point(0, 0));
}
```

Indem Sie der `copyPixels`-Methode als sechsten Parameter `true` übergeben, wird diese angewiesen, `brush_bd` (vierter Parameter) als Alphamaske zu nutzen. Nun passiert Folgendes: Für den Kopiervorgang der Pixeldaten von `brushed_part_bd` in `output_bd` werden die Alphawerte von `brush_bd` herangezogen. Somit werden Flächen mit einem Alphawert von 0% (alle Bildteile außerhalb des schwarzen Kreises) nicht kopiert und Bitmapdaten mit einem Alphawert von 20% (also der sichtbare schwarze Kreis) mit einer Stärke von 20% kopiert. So entsteht die kreisförmige Verwischung.

In der zweiten Zeile schließlich wird der zu kopierende Bildteil aus `output_bd` in `brushed_part_bd` gespeichert. Hierzu wird `draw_rectangle` (also das Quadrat, über dem sich die Maus befindet) als Positionsangabe genutzt.

Aber – wird jetzt mancher fragen – wie soll denn in der ersten Zeile in Schritt 10 `brushed_part_bd` kopiert werden, wenn dieses Objekt erst in der folgenden Zeile erstellt wird? Antwort: Würde man den Code der beiden Zeilen tauschen, würde sich das Bild nicht verändern – es würden nur unveränderte Bildteile kopiert. Darum nutzt man eine Verzögerung von

Schritt 10

Bilddaten kopieren

Schritt 11

Logik

einem Durchlauf. Wenn `onBrushMove()` also das erste Mal ausgeführt wird (bei der ersten Mausbewegung), verändert sich das Bild tatsächlich nicht, weil `brushed_part_bd` noch leer ist. Da der Nutzer aber mehr als eine Bewegung mit der Maus macht, bleibt dies unbemerkt.

Im fertigen Flash-Film können Sie nun mit gedrückter linker Maustaste Bildbereiche verwischen.



Abbildung 7 ►
Mit der Maus kann das Bild verwischt werden.

Der Eindruck des Verwischens entsteht, weil wir die Bilddaten nur mit einem Wert von 20 % kopieren. Setzen Sie den Wert von `alphaMultiplifier` (Zeile 3 in Schritt 5) doch einmal auf 0.9 (100 % Alpha), und schauen Sie sich das Ergebnis an.

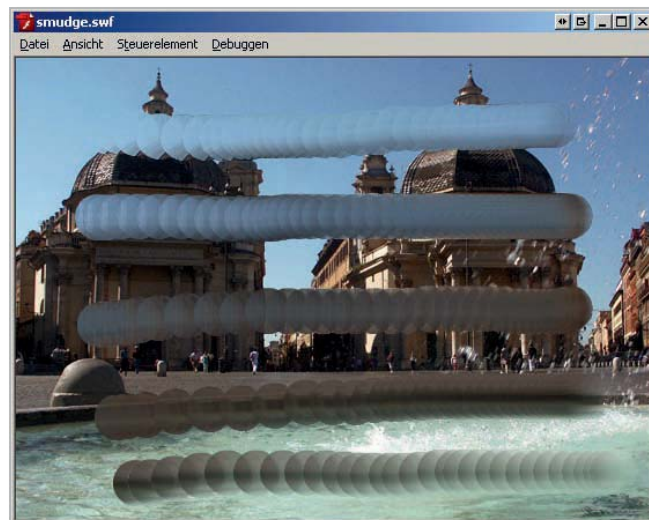


Abbildung 8 ►
Der Verwischen-Effekt kann individuell eingestellt werden.

Vergrößerungs-Effekt

Bildteile wie mit einer echten Lupe vergrößern

Schon seit Flash 5 kann man einzelne Bildteile mit Maskentricks vergrößern. Seit Flash 8 ist dies nun sehr performant und mit weniger Code möglich. In diesem Workshop lernen Sie, wie Sie eine Lupe in Flash erstellen und was dabei zu beachten ist.

Zielsetzungen:

Bildteile mit der BitmapData-Klasse vergrößern

Vergrößerten Bildteil mit einer runden Maske maskieren

Lupe mit einem Glow-Filter versehen

[Ergebnis: 09_VERGRÖßERUNGS-EFFEKT • PIXELZOOM.SWF]

```
import flash.display.BitmapData;
import flash.geom.Matrix;
import flash.geom.Rectangle;
import flash.filters.GlowFilter;
function drawCircle(mc:MovieClip, x:Number, y:Number,
r:Number):Void {
    mc.moveTo(x+r, y);
    mc.curveTo(r+x, Math.tan(Math.PI/8)*r+y, Math.sin(Math.
PI/4)*r+x, Math.sin(Math.PI/4)*r+y);
    mc.curveTo(Math.tan(Math.PI/8)*r+x, r+y, x, r+y);
    mc.curveTo(-Math.tan(Math.PI/8)*r+x, r+y, -Math.sin(Math.
PI/4)*r+x, Math.sin(Math.PI/4)*r+y);
    mc.curveTo(-r+x, Math.tan(Math.PI/8)*r+y, -r+x, y);
    mc.curveTo(-r+x, -Math.tan(Math.PI/8)*r+y, -Math.sin(Math.
PI/4)*r+x, -Math.sin(Math.PI/4)*r+y);
    mc.curveTo(-Math.tan(Math.PI/8)*r+x, -r+y, x, -r+y);
    mc.curveTo(Math.tan(Math.PI/8)*r+x, -r+y, Math.sin(Math.
PI/4)*r+x, -Math.sin(Math.PI/4)*r+y);
    mc.curveTo(r+x, -Math.tan(Math.PI/8)*r+y, r+x, y);
}
```



Schritt 1

Dokument erstellen

Erstellen Sie ein neues AS2-Flash-Dokument, und öffnen Sie mit **[Strg]/[⌘] + [J]** die Dokumenteigenschaften. Stellen Sie die Größe 550x400 Pixel und eine BILDRATE von 25 bps ein. HINTERGRUNDFARBE und sonstige Einstellungen spielen keine Rolle.



Abbildung 1 ►
In den Dokumenteigenschaften legen Sie die Grundeinstellungen für Ihren Flash-Film fest.

Schritt 2

Bild importieren

Importieren Sie über **DATEI • IMPORTIEREN • IN BÜHNE IMPORTIEREN** oder mit **[Strg]/[⌘] + [R]** die Datei ROM.JPG. Die Größe dieses Bildes entspricht mit 550x400 Pixel der Bühnengröße.

Markieren Sie das importierte Bild auf der Bühne, und rufen Sie über einen Klick auf die rechte Maustaste bzw. **[Ctrl]**-Klick das Kontextmenü auf. Dort wählen Sie den Eintrag **IN SYMBOL KONVERTIEREN**.

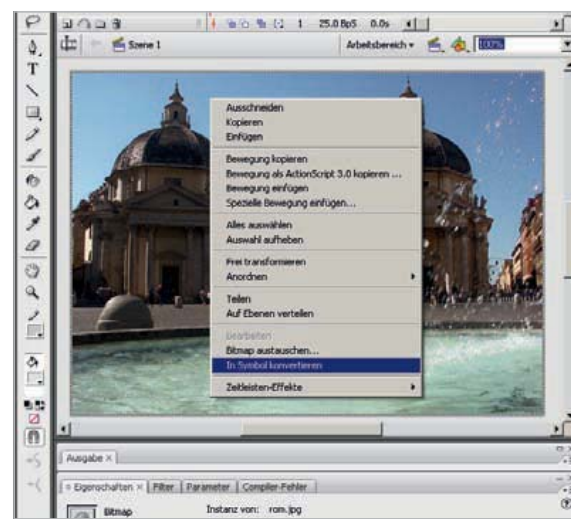


Abbildung 2 ►
Per Klick auf die rechte Maustaste erreichen Sie den Menüpunkt **IN SYMBOL KONVERTIEREN**.

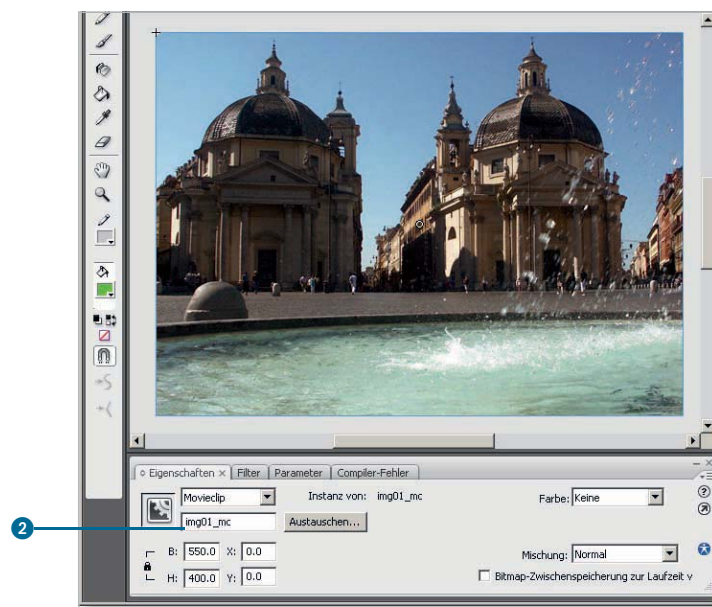
Wählen Sie in dem sich nun öffnenden Dialog die Option MOVIECLIP und geben Sie ihm den Symbolnamen »img01_mc« ❶. Bestätigen Sie Ihre Eingaben anschließend mit OK.



◀ **Abbildung 3**

Das Bild wird in einen Movieclip mit dem Namen »img01_mc« konvertiert.

Anschließend vergeben Sie im Eigenschaftenfenster den Instanznamen »img01_mc« ❷.



◀ **Abbildung 4**

Geben Sie dem Movieclip den Instanznamen »img01_mc«.

Zunächst müssen Sie verschiedene ActionScript-Klassen für unser Skript verfügbar machen. Rufen Sie das AKTIONEN-Fenster über FENSTER N AKTIONEN oder **[F9]** bzw. **[Alt] + [F9]** im ersten Schlüsselbild auf, und kopieren Sie folgenden Code hinein:

```
import flash.display.BitmapData;
import flash.geom.Matrix;
import flash.geom.Rectangle;
import flash.filters.GlowFilter;
```

Schritt 3 Skript vorbereiten

Schritt 4

Kreisfunktion bereitstellen

Im Verlauf des Workshops werden wir eine Funktion zum Zeichnen eines Kreises benötigen. Fügen Sie Ihrem Skript folgende Zeilen hinzu:

```
function drawCircle(mc:MovieClip, x:Number, y:Number,
r:Number):Void {
    mc.moveTo(x+r, y);
    mc.curveTo(r+x, Math.tan(Math.PI/8)*r+y, Math.sin(Math.
PI/4)*r+x, Math.sin(Math.PI/4)*r+y);
    mc.curveTo(Math.tan(Math.PI/8)*r+x, r+y, x, r+y);
    mc.curveTo(-Math.tan(Math.PI/8)*r+x, r+y, -Math.sin(Math.
PI/4)*r+x, Math.sin(Math.PI/4)*r+y);
    mc.curveTo(-r+x, Math.tan(Math.PI/8)*r+y, -r+x, y);
    mc.curveTo(-r+x, -Math.tan(Math.PI/8)*r+y, -Math.sin(Math.
PI/4)*r+x, -Math.sin(Math.PI/4)*r+y);
    mc.curveTo(-Math.tan(Math.PI/8)*r+x, -r+y, x, -r+y);
    mc.curveTo(Math.tan(Math.PI/8)*r+x, -r+y, Math.sin(Math.
PI/4)*r+x, -Math.sin(Math.PI/4)*r+y);
    mc.curveTo(r+x, -Math.tan(Math.PI/8)*r+y, r+x, y);
}
```

Diese Funktion ist der Flash-Hilfe entnommen. Sie finden sie, wenn Sie in der Hilfe nach `curveTo` suchen. Zu ihrer Anwendung erfahren Sie mehr in Schritt 7 dieses Workshops.

Schritt 5

Lupen-Movieclip erstellen

Sie werden im Laufe dieses Workshops verschiedene Movieclips und BitmapData-Objekte erstellen, die alle einen »Mutter«-Movieclip benötigen. Dieser »Mutter«-Movieclip wird später dynamisch, je nach Mausposition, auf der Bühne bewegt.

Fügen Sie Ihrem Skript ganz am Ende folgende Codezeilen hinzu, um diesen Container-Movieclip zu erstellen:

```
var lens_mc:MovieClip = this.createEmptyMovieClip("lens_mc",
this.getNextHighestDepth());
lens_mc._x = 150;
lens_mc._y = 100;
```

In der ersten Zeile wird ein Movieclip namens `lens_mc` erstellt, der in den beiden folgenden Zeilen auf die x-Position 150 Pixel und die y-Position 100 Pixel positioniert wird.

Ergänzen Sie das Skript wieder ganz am Ende nun um folgende Zeilen:

```
var zoom_mc:MovieClip = lens_mc.createEmptyMovieClip  
("zoom_mc", 1);  
var mask_mc:MovieClip = lens_mc.createEmptyMovieClip  
("mask_mc", 2);
```

Innerhalb des `lens_mc`-Movieclips erstellen Sie in der ersten Zeile den Movieclip `zoom_mc`. Er wird später als Container für den vergrößerten Bildteil dienen.

Der nächste Movieclip `mask_mc` in der zweiten Zeile ist die Maske, die dafür sorgt, dass ein kreisrunder Bildteil ausgegeben wird. So entsteht später der Eindruck einer Lupe.

Nun werden Sie einen Kreis in dem zuvor erstellten `mask_mc`-Movieclip zeichnen und die Maske auf den ebenfalls erstellten `zoom_mc`-Movieclip anwenden.

Fügen Sie Ihrem Skript ganz unten folgende Zeilen hinzu:

```
mask_mc.beginFill(0x000000);  
drawCircle(mask_mc, 50, 50, 50);
```

In der ersten Zeile legen Sie fest, dass die Farbe des Kreises Schwarz sein soll. Die Farbe spielt eigentlich keine Rolle, da Sie diesen Movieclip ja als Maske anwenden werden und der User die Farbe somit nicht zu Gesicht bekommt.

Trotzdem geben wir eine Farbe an, damit wir, wenn wir den Flash-Film im nächsten Schritt testen, etwas sehen.

Um den Kreis zu zeichnen, greifen Sie in der zweiten Zeile auf die in Schritt 4 erstellte Funktion `drawCircle()` zurück. Als ersten Parameter übergeben Sie den Movieclip, in den gezeichnet werden soll. Parameter 2 und 3 legen den Mittelpunkt über die x- und y-Position des zu zeichnenden Kreises fest. Als vierten Parameter schließlich geben Sie an, welchen Radius der Kreis haben soll.

Exportieren Sie Ihren Flash-Film nun testweise über / +  oder über STEUERUNG • FILM TESTEN. Sie sollten nun einen schwarzen Kreis sehen.

Schritt 6

Vergrößerungs- und Masken-Movieclip erstellen

Schritt 7

Maske zeichnen und anwenden

Abbildung 5 ►
Die Maske ist angelegt und muss nun noch angewendet werden.



Mit der nächsten Zeile wenden Sie diesen Kreis als Maske auf den zurzeit noch leeren Movieclip `zoom_mc` an:

```
zoom_mc.setMask(mask_mc);
```

Wenn Sie den Film jetzt erneut testen, werden Sie sehen, dass der schwarze Kreis verschwunden ist. Da sich in dem Movieclip `zoom_mc` noch keine Bildinformationen befinden, gibt es noch nichts zu maskieren. Später, wenn Bilddaten in `zoom_mc` dargestellt werden, werden diese korrekt mit der Maske `mask_mc` maskiert.

Schritt 8

Glow-Filter erstellen und anwenden

Um die Lupe später besser von dem Ausgangsbild unterscheiden zu können, werden Sie einen Glow-Filter auf den Lupen-Movieclip anwenden. Ergänzen Sie Ihr Skript ganz am Ende um folgende Codezeilen:

```
var filter:GlowFilter = new GlowFilter(0x000000, 50, 20, 20, 3);
zoom_mc.filters = [filter];
```

Zunächst erstellen Sie einen Glow-Filter mit der Farbe Schwarz, einem Alphawert von 50, einer horizontalen und vertikalen Weichzeichnung von 20 und einer Stärke von 3. Spielen Sie ruhig ein bisschen mit den Parametern herum. Der Filter ist in der Flash-Hilfe ausführlich und gut erklärt.

In der zweiten Zeile schließlich wenden Sie den Filter auf den Movieclip `zoom_mc` an.

Schritt 9

Abbild des Originalbildes erstellen

In späteren Schritten werden wir Teile des Originalbildes kopieren und vergrößert darstellen. Um Bildteile aus dem Ausgangsbild »img01_mc« auf der Bühne kopieren zu können, muss das Bild zunächst als BitmapData-

Objekt vorliegen. Fügen Sie Ihrem Skript dazu wieder ganz am Ende folgende Zeilen hinzu:

```
var original_bmp:BitmapData = new BitmapData(img01_mc._width,  
    img01_mc._height, true);  
original_bmp.draw(img01_mc);
```

In der ersten Zeile erstellen Sie ein leeres BitmapData-Objekt mit den Maßen des auf der Bühne importierten Bildes. Danach zeichnen Sie ein Abbild des Movieclips in das erstellte BitmapData-Objekt.

Um Bitmapdaten vergrößern zu können, muss Flash genaue Anweisungen bekommen, wie dies geschehen soll. Ergänzen Sie folgende Codezeilen am Ende Ihres Skripts:

```
var zoom_matrix:Matrix = new Matrix();  
zoom_matrix.scale(2, 2);
```

Das Werkzeug dafür ist das Matrix-Objekt. Mit dem Matrix-Objekt lässt sich sehr genau (und unglaublich komplex) beschreiben, wie Bitmapdaten verändert werden sollen. In unserem Fall ist es recht einfach: In der ersten Zeile erstellen Sie ein leeres Matrix-Objekt, dem in der folgenden Zeile mitgeteilt wird, dass es um den x- und y-Faktor 2 vergrößert werden soll. Unsere Lupe wird also um den Faktor 2 vergrößern – das Bild in der Lupe wird somit doppelt so groß erscheinen wie das Ausgangsbild.

In diesem Schritt erstellen Sie das BitmapData-Objekt, das die vergrößerten Bilddaten enthalten wird. Ergänzen Sie Ihr Skript wie folgt:

```
var zoom_bmp:BitmapData = new BitmapData(100, 100, false,  
    0xFF0000);  
zoom_mc.attachBitmap(zoom_bmp, 1);
```

Zuerst erstellen Sie ein BitmapData-Objekt mit der Größe 100 x 100 Pixel. Dies entspricht der Kantenlänge der zuvor erstellten Kreismaske. Außerdem soll das Objekt keine Transparenzen unterstützen (`false`) und zunächst rot sein (`0xFF0000`).

Schritt 10

Skalierungsmatrix erstellen

Schritt 11

BitmapData-Objekt für das vergrößerte Bild

Nun fügen Sie das BitmapData Objekt `zoom_bmp` an den Movieclip `zoom_mc` an. Dadurch werden alle Änderungen, die ab jetzt an `zoom_bmp` vorgenommen werden, automatisch in `zoom_mc` gezeichnet.

Testen Sie nun Ihr Skript. Sie werden einen roten Kreis sehen, der von einem schwarzen Glühen umhüllt ist.

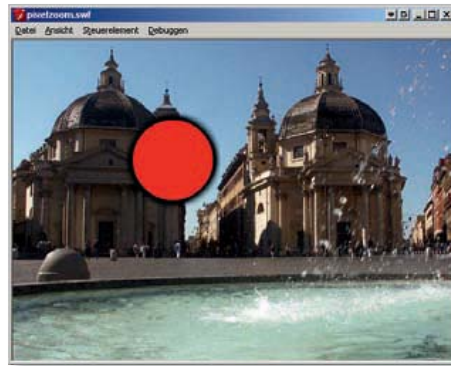


Abbildung 6 ►

Die Lupe hebt sich nun deutlich von dem Hintergrundbild ab.

Schritt 12

Bildteil bei Mausbewegung vergrößern

Jetzt wird die Lupe zum Leben erweckt; denn bisher sieht man statt einer Vergrößerung nur einen roten Kreis. Dafür benötigen Sie zunächst eine Funktion, die immer ausgeführt wird, wenn der User die Maus bewegt. Ergänzen Sie Ihr Skript um folgende Zeilen:

```
1: onMouseMove = function(){
2:     var copy_bmp:BitmapData = new BitmapData(50, 50, false);
3:     copy_bmp.copyPixels(original_bmp, new Rectangle(_xmouse -
4:         25, _ymouse - 25, 50, 50), new Point(0, 0));
5:     zoom_bmp.draw(copy_bmp, zoom_matrix);
```

In Zeile 2 erstellen Sie ein leeres BitmapData-Objekt mit der Kantenlänge von 50 x 50 Pixel. Dies ist der zu vergrößernde Bildteil.

Nun werden Sie die Bilddaten kopieren. In Zeile 3 nutzen Sie die `copyPixels`-Methode um Bilddaten aus dem zuvor erstellten `original_bmp` in Ihr eben erstelltes `copy_bmp` zu kopieren.

Als zweiten Parameter übergeben Sie ein Rectangle-Objekt, das den Bildausschnitt des zu kopierenden Bildbereiches festlegt. Da die Lupenmitte (und nicht die obere linke Ecke) über der Maus kopiert werden soll, müssen Sie von den Parametern 1 und 2 des Rectangle-Objekts (x- und y-Position) jeweils die Hälfte abziehen, also 25. Der dritte und vierte Parameter des Rectangle-Objekts legen die Breite und Höhe fest und entsprechen den Maßen des zuvor erstellten Bitmap-Objekts.

Mit dem Point-Objekt, das Sie der `copyPixels`-Methode als dritten Parameter übergeben, geben Sie an, an welcher Position (x und y) die kopierten Bilddaten im Ziel-Bitmap-Objekt (`copy_bmp`) eingefügt werden sollen.

Um den kopierten Bildteil schließlich anzuzeigen, zeichnen Sie das gerade erstellte BitmapData-Objekt `copy_bmp` in das in Schritt 11 dieses Workshops erstellte BitmapData-Objekt `zoom_bmp`. Indem Sie als zweiten Parameter `zoom_matrix` übergeben, sorgen Sie dafür, dass `copy_bmp` beim Zeichenvorgang um den Faktor 2 in x- und y-Richtung vergrößert wird.

Zum Schluss müssen Sie noch dafür sorgen, dass die Lupe der Maus folgt. Fügen Sie Ihrem Code folgende Zeilen hinzu:

```
lens_mc._x = _xmouse - 50;  
lens_mc._y = _ymouse - 50;  
}
```

Immer wenn der User die Maus bewegt, wird der Movieclip `lens_mc` auf die Position der Maus verschoben. Damit sich die Lupenmitte unter dem Mauszeiger befindet, müssen Sie von der x- und y-Position die Hälfte der Kantenlänge, also jeweils 50 Pixel, abziehen.

Wenn der Flash-Film aufgerufen wird, muss der User ja noch nicht zwangsläufig die Maus bewegt haben – diese kann sich ja auch außerhalb des Flash-Films befinden. Darum rufen wir die Methode `onMouseMove()` einmal »per Hand« am Ende unseres Skripts auf.

```
onMouseMove();
```



Schritt 13

Mausverfolgung der Lupe

Schritt 14

Startposition festlegen

◀ Abbildung 7

Die Lupe folgt der Mausbewegung und vergrößert das dahinter liegende Bild.

Farbwähler

Farbwerte eines Bildes auslesen und anzeigen

```
import flash.display.BitmapData;
var display_mc:MovieClip = this.createEmptyMovieClip(
    "display_mc", this.getNextHighestDepth());
display_mc._x = 10;
display_mc._y = 10;
var bg_mc:MovieClip = display_mc.createEmptyMovieClip(
    "bg_mc", 1);
bg_mc.beginFill(0xDDDDDD);
bg_mc.lineStyle(1, 0x666666);
bg_mc.moveTo(0, 0);
bg_mc.lineTo(100, 0);
bg_mc.lineTo(100, 100);
bg_mc.lineTo(0, 100);
bg_mc.lineTo(0, 0);
bg_mc.endFill();
```

Seit Flash 8 ist endlich der direkte Zugriff auf Pixelinformationen innerhalb von Flash möglich. In diesem Workshop werden Sie lernen, die Farbwerte eines Ausgangsbildes auszulesen und sowohl die Farbe als auch ihren Farbwert anzuzeigen.

Zielsetzungen:

- Farbwerte auslesen
- Textfeld erstellen
- Aufgenommene Farbe darstellen
- Farbwert in die Zwischenablage kopieren

[Ergebnis: 10_FARBWAehler • COLORPICKER.SWF]



Erstellen Sie ein neues AS2-Flash-Dokument, und öffnen Sie mit **[Strg]/[⌘] + [J]** die Dokumenteigenschaften. Stellen Sie die Größe 550x400 Pixel und eine BILDRATE von 25 bps ein. HINTERGRUNDFARBE und sonstige Einstellungen spielen keine Rolle.



Schritt 1

Dokument erstellen

◀ **Abbildung 1**
Dokumenteigenschaften festlegen

Importieren Sie über **DATEI • IMPORTIEREN • IN BÜHNE IMPORTIEREN** oder über **[Strg]/[⌘] + [R]** die Datei ROM.JPG. Sie hat bereits eine Größe von 550 x 400 Pixel, was der Bühnengröße entspricht.

Markieren Sie das importierte Bild auf der Bühne, und konvertieren Sie es über **[F8]** in einen Movieclip. Geben Sie dem Movieclip in dem sich nun öffnenden Dialog einen beliebigen Symbolnamen und bestätigen Sie mit OK.



Schritt 2

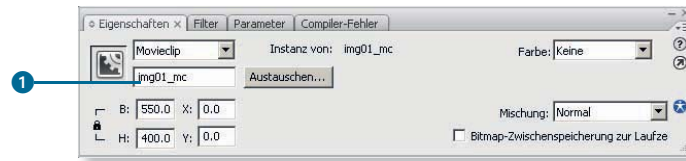
Bild importieren

◀ **Abbildung 2**
Konvertieren Sie das Bild in ein Symbol.

Anschließend geben Sie dem Movieclip auf der Bühne im Eigenschaftsfenster den Instanznamen »img01_mc« **1**.

Abbildung 3 ►

Geben Sie dem Movieclip
den Instanznamen
»img01_mc«.



Schritt 3

Skript vorbereiten

Zunächst müssen Sie die BitmapData-Klasse für Ihr Skript verfügbar machen. Markieren Sie das erste Schlüsselbild, und rufen Sie das AKTIONEN-Fenster über FENSTER • AKTIONEN oder **[F9]** bzw. **[Alt] + [F9]** auf. Kopieren Sie folgenden Code hinein:

```
import flash.display.BitmapData;
```

Schritt 4

Ausgabe-Movieclip erstellen

Als Erstes erstellen Sie einen Movieclip, der als Container für das Farbfeld und das Textfeld dienen wird. Ergänzen Sie folgende Zeilen direkt nach der Import-Anweisung:

```
var display_mc:MovieClip = this.createEmptyMovieClip("display_
mc", this.getNextHighestDepth());
display_mc._x = 10;
display_mc._y = 10;
```

In der ersten Zeile erstellen Sie einen Movieclip, den Sie dann in den beiden folgenden Zeilen etwas von der linken, oberen Ecke versetzen. Sie können hier natürlich auch eine beliebige andere Position auf der Bühne wählen.

Schritt 5

Hintergrund erstellen

Damit sich Ihr Ausgabe-Movieclip auch von dem Bild »img01_mc« (aus dem die Farbwerte aufgenommen werden) grafisch abhebt, erstellen Sie einen Hintergrund-Movieclip.

Erweitern Sie Ihr Skript um folgende Zeilen:

```
1: var bg_mc:MovieClip = display_mc.createEmptyMovieClip
   ("bg_mc", 1);
2: bg_mc.beginFill(0xDDDDDD);
3: bg_mc.lineStyle(1, 0x666666);
4: bg_mc.moveTo(0, 0);
5: bg_mc.lineTo(100, 0);
6: bg_mc.lineTo(100, 100);
```

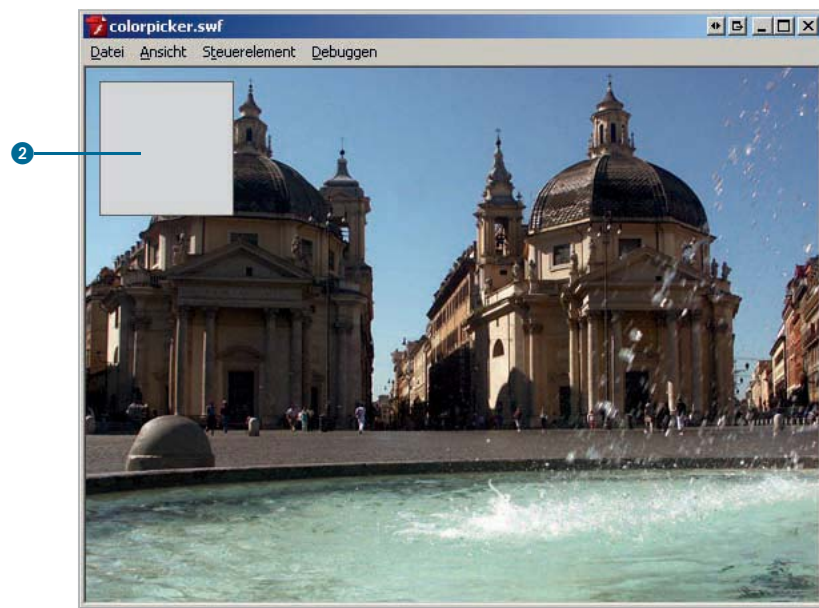
```

7:  bg_mc.lineTo(0, 100);
8:  bg_mc.lineTo(0, 0);
9:  bg_mc.endFill();

```

Zunächst erstellen Sie einen leeren Movieclip. In den folgenden Zeilen zeichnen Sie ein graues Quadrat (Zeile 2) mit einer Linienstärke von 1 in einem etwas dunkleren Grau (Zeile 3) und einer Kantenlänge von 100 x 100 Pixel (Zeilen 4 bis 9).

Wenn Sie Ihr Dokument nun über `Strg/⌘ + ⏏` oder **STEUERUNG • FILM TESTEN** veröffentlichen, sehen Sie die graue Hintergrundfläche **2**.



Flash-Hilfe

Weitere Informationen, wie Sie Pfade in Movieclips zeichnen können, finden Sie ausführlich und gut beschrieben in der Flash-Hilfe.

◀ Abbildung 4

Das Feld für die Anzeige der Farbwerte ist einsatzbereit.

Fügen Sie Ihrem Skript nun folgende Zeilen hinzu:

```

1:  var color_mc:MovieClip = display_mc.
    createEmptyMovieClip("color_mc", 2);
2:  color_mc._x = 1;
3:  color_mc._y = 1;

```

In Zeile 1 erstellen Sie wiederum einen leeren Movieclip. Damit das Farbfeld, das Sie im nächsten Schritt zeichnen werden, nicht auf der zuvor gezeichneten Außenlinie von `bg_mc` liegt, verschieben Sie den Movieclip in den Zeilen 2 und 3 um jeweils ein Pixel. So entsteht ein kleiner Abstand zur Außenlinie.

Schritt 6

Farb-Movieclip erstellen

Schritt 7

BitmapData-Objekt für die Darstellung der Farbe

Nun erstellen Sie das BitmapData-Objekt, das später die Farbe des ausgelesenen Pixels darstellt. Ergänzen Sie Ihr Skript um folgende Zeilen:

```
var color_bmp:BitmapData = new BitmapData(99, 80, false,  
    0xFF0000);  
color_mc.attachBitmap(color_bmp, 1);
```

Wiederum sorgen Sie dafür, dass das Farbfeld nicht über der Außenlinie von `bg_mc` liegt, indem Sie das BitmapData-Objekt nur 99 Pixel breit erstellen. Um unter dem Farbfeld noch Platz für das Textfeld zu lassen, geben Sie die Höhe mit 80 Pixel an. Außerdem soll Ihr BitmapData-Objekt keine Transparenzen unterstützen (`false`) und die Ausgangsfarbe Rot haben (`0xFF0000`).

BitmapData-Objekte existieren nur im Speicher. Um sie anzuzeigen, müssen Sie einem Movieclip angefügt werden. Sie tun dies in der zweiten Zeile: Sie fügen das BitmapData-Objekt dem in Schritt 6 erstellten Farb-Movieclip an.

Veröffentlichen Sie Ihr Dokument nun testweise über `Strg/⌘ + ↵` oder STEUERUNG • FILM TESTEN.

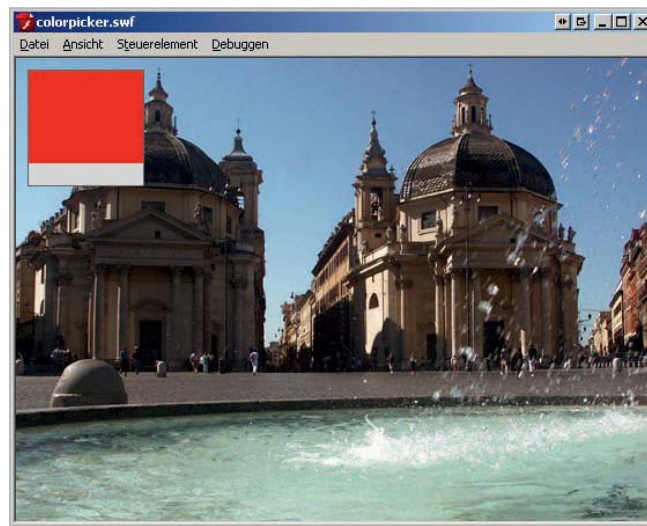


Abbildung 5 ►
Über der grauen Hintergrundfläche wird nun eine zweite, rote Fläche angezeigt.

Schritt 8

Textfeld für den Farbwert

Nun erstellen Sie das Textfeld, in dem der Farbwert des aufgenommenen Pixels dargestellt wird. Erweitern Sie Ihr Skript wie folgt:

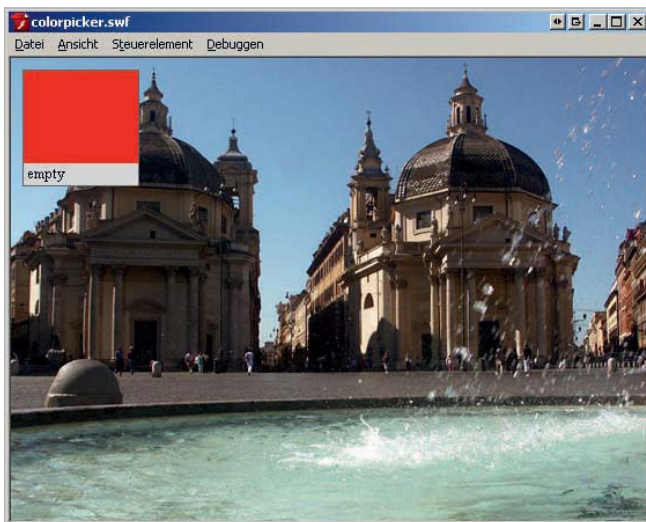
```
1: var color_txt:TextField = display_mc.createTextField  
    ("color_txt", 3, 3, 80, 80, 20);
```

```
2: color_txt.text = "empty";
3: color_txt.selectable = false;
```

Innerhalb von `display_mc` erstellen Sie in der ersten Zeile ein Textfeld mit dem Namen `color_txt`, das eine Tiefe von 3 (Parameter 2) hat und auf der x-Position von 3 Pixel erstellt wird (Parameter 3). Damit das Textfeld unter dem Farbausgabe-Movieclip erscheint, geben Sie ihm eine y-Position von 80 Pixel (Parameter 4). Schließlich legen Sie eine Breite von 80 und eine Höhe von 20 Pixel fest (Parameter 5 und 6).

In Zeile 2 geben Sie an, dass zunächst der Text »empty« angezeigt werden soll. Außerdem soll das Textfeld nicht auswählbar sein (Zeile 3).

Veröffentlichen Sie Ihr Dokument nun über `[Strg]/[⌘] + [↵]`. Unterhalb der roten Fläche wird nun ein Textfeld mit der Zeichenkette »empty« angezeigt.



Zeichenketten

Eine Zeichenkette oder ein String (englisch) ist eine Folge von Zeichen (z. B. Buchstaben, Ziffern, Sonderzeichen und Steuerzeichen) aus einem definierten Zeichensatz. Zeichen können sich in einer Zeichenkette wiederholen, die Reihenfolge der Zeichen ist definiert. Zeichenketten sind somit Sequenzen aus Symbolen mit endlicher Länge. (Quelle: *wikipedia.de*)

◀ Abbildung 6

Das Textfeld wurde korrekt angelegt und zeigt zunächst noch den von Ihnen definierten Standardwert »empty«.

Um auf Pixelinformationen zugreifen zu können, benötigen Sie ein `BitmapData`-Objekt. Flash kann nicht direkt auf Pixeldaten von Movieclips oder Bildern auf der Bühne zugreifen. Dieses `BitmapData`-Objekt erstellen Sie nun. Fügen Sie folgende Zeilen hinzu:

```
var color_canvas_bmp:BitmapData = new BitmapData(img01_mc.
    _width, img01_mc._height, false);
color_canvas_bmp.draw(img01_mc);
```

Dazu erstellen Sie zunächst ein leeres `BitmapData`-Objekt mit den Maßen des Ausgangsbildes auf der Bühne (»img01_mc«). In der zweiten Zeile

Schritt 9

Abbild des Ausgangsbildes erstellen

schließlich zeichnen Sie ein Abbild von »img01_mc« in Ihr BitmapData-Objekt. Von nun an können Sie die Farbwerte aus diesem Objekt auslesen.

Schritt 10

Farbwert bei Mausbewegung auslesen

Das Skript soll den Farbwert bei jeder Mausbewegung des Users auslesen. Ergänzen Sie Ihr Skript um folgende zwei Codezeilen:

```
this.onMouseMove = function(){  
    var c:Number = color_canvas_bmp.getPixel(_xmouse, _ymouse);
```

Die erste Zeile sorgt dafür, dass der in dieser Funktion enthaltene Code immer ausgeführt wird, wenn der User die Maus bewegt.

In der zweiten Zeile lesen Sie dann den Farbwert aus. Dazu überprüfen Sie in dem BitmapData-Objekt `color_canvas_bmp` (das ein Abbild des Ausgangsbildes enthält) das Pixel, über dem sich die Maus auf der Bühne (also über »img01_mc«) gerade befindet. Die Funktion gibt eine Zahl zwischen 0 und 16,7 Millionen zurück.

Schritt 11

Farbausgabefeld färben

Ergänzen Sie Ihr Skript wie folgt:

```
color_bmp.fillRect(color_bmp.rectangle, c);
```

Jetzt füllen Sie das zuvor erstellte Farbausgabefeld mit der eben ausgelesenen Farbe. Dazu übergeben Sie der `fillRect`-Methode zunächst das Quadrat, das befüllt werden soll – in unserem Fall die komplette Fläche von `color_bmp`.

Als zweiten Parameter übergeben Sie die eigentliche Farbe, die Sie zuvor in der Variablen `c` gespeichert haben.

Schritt 12

Farbwert als Zeichenkette ausgeben

Unser Ausgabe-Movieclip soll nicht nur die Farbe darstellen, sondern auch den Farbwert als Text anzeigen.

Fügen Sie daher folgenden Code ein:

```
color_txt.text = "0x"+c.toString(16).toUpperCase();
```

Mit dieser Zeile geben Sie den aufgenommenen Farbwert als Zeichenkette aus. Das `"0x"` stellen Sie voran, damit man sofort sieht, dass es sich um einen Farbwert handelt.

Mit der Methode `toString(16)` konvertieren Sie den Farbwert vom Dezimal- ins Hexadezimalsystem. Anschließend sorgen Sie mit `toUpperCase()` dafür, dass alle Buchstaben als Großbuchstaben angezeigt werden.

Im letzten Schritt werden Sie jetzt die Zwischenablage füllen:

```
System.setClipboard(c.toString(16));  
}
```

Nun wird immer, wenn der User die Maus bewegt, die Zwischenablage mit dem aufgenommenen Farbwert als Hexadezimalzahl gefüllt. Sie können jetzt (beispielsweise) in Photoshop wechseln und den Farbwert in den Farbwähler einfügen.



▲ **Abbildung 7**

Farbe und Farbwert des Bildes an der aktuellen Position des Mauszeigers werden angezeigt.

Schritt 13

Zwischenablage füllen

Rückgabewerte

Funktionen oder Methoden dienen oft der Berechnung eines Wertes. Der Methode werden Werte als Parameter übergeben, die die Methode zur Berechnung nutzt. Anschließend gibt die Funktion den fertig berechneten Wert als Rückgabewert zurück. Die Rückgabewerte der Flash-eigenen Funktionen können Sie der Flash-Hilfe entnehmen.

Anzeigeliste und attachMovie()

Komplizierter?

Was auf den ersten Blick komplizierter erscheint, ist im praktischen Einsatz tatsächlich einfacher. Wenn Sie `attachMovie()` bereits kennen und sich jetzt erstmals mit der Anzeigeliste befassen, werden Sie sehr bald schon die enormen Vorteile der Anzeigeliste zu schätzen wissen.

Was ist eine Anzeigeliste, was attachMovie()?

Eine in Flash immer wieder auftretende Aufgabenstellung ist es, Objekte per ActionScript auf die Bühne zu holen. Wobei der Begriff »Bühne« hier im technischen Sinne nicht wortwörtlich zu verstehen ist – Sie können Objekte ebenso einem auf der Bühne befindlichen Objekt anhängen. Ein Movieclip beispielsweise kann beliebig viele untergeordnete Movieclips beinhalten; diese wiederum können weitere Movieclips beinhalten usw. Daraus ergibt sich eine Hierarchie, die man sich als Baumstruktur vorstellen kann. Die Methode, mit der man bis ActionScript 2 Movieclips an andere Movieclips oder an die Bühne angefügt hat, lautet `attachMovie()`. In ActionScript 3 gibt es dafür keine einzelne Funktion mehr. Vielmehr kommt ein ausgefeiltes Konzept zum Einsatz – die Anzeigeliste.

attachMovie()

Die Funktion `attachMovie()` fügt der Bühne oder einem Movieclip einen Movieclip an. So kann Stück für Stück die hierarchische Struktur der dargestellten Objekte aufgebaut werden. Der folgende Code platziert einen Movieclip auf der Bühne:

```
_root.attachMovie("mc", "mc_instanz", 1);
```

Hierbei steht der erste Parameter `"mc"` für den Bezeichner, also den Namen, unter dem Sie den anzufügenden Movieclip mit ActionScript erreichen können. Dieser Bezeichner kann nur über die Bibliothek vergeben werden, indem Sie mit der rechten Maustaste auf das entsprechende Symbol klicken und unter VERKNÜPFUNG den Bezeichner angeben. Der zweite Parameter gibt den Instanznamen an. Als dritter Parameter wird die Tiefe angegeben.

Tiefenverwaltung und attachMovie()

ActionScript 1 und 2 verfügen über ein lineares Tiefenverwaltungssystem, bei dem Werte zwischen -16384 und 1048575 als Tiefe zulässig sind. Ein Objekt mit einer geringeren Tiefe wird von einem Objekt mit höherer

Tiefe verdeckt. Jeder Tiefenwert kann nur einmal vergeben werden. Verwenden Sie beim Erstellen oder Anfügen von Objekten zum zweiten Mal eine bereits vergebene Tiefe, wird das ursprüngliche Objekt überschrieben. Angenommen, Sie hätten den Beispielcode aus Schritt 1.1.1 in einem Flash-Projekt verwendet und Sie möchten jetzt einen weiteren Movieclip auf die Bühne holen. Dazu nutzen Sie folgenden Code:

```
_root.attachMovie("mc", "mc_instanz2", 1);
```

Dieser Code würde zwar eine neue Instanz des verknüpften Movieclips erzeugen, dabei jedoch die bereits vorhandene löschen, da die Tiefe 1 bereits vergeben ist. Die Tiefenverwaltung bleibt Ihnen so im Grunde selbst überlassen. Sie müssen selbst dafür Sorge tragen, dass bereits vergebene Tiefen nicht aus Versehen überschrieben werden. Zwar bietet ActionScript einige Funktionen an, die den Umgang mit den linearen Tiefen erleichtern. So ermittelt beispielsweise die Funktion `getNextHighestDepth()` die nächsthöchste noch nicht vergebene Tiefe und stellt somit sicher, dass kein Objekt versehentlich überschrieben wird. Dennoch ist die Tiefenverwaltung in ActionScript 1 und 2 ein heikles Thema, das schon viele Nerven gekostet hat.

Die Anzeigeliste

Seit ActionScript 3 gibt es die Anzeigeliste, die viele Verbesserungen mit sich bringt. So kann etwa eine bessere Performance erreicht werden, indem die Beschränkung auf Movieclips als hierarchiefähige Anzeigeobjekte aufgehoben wurde. Ein Movieclip verfügt immer über eine eigene Zeitleiste, auch wenn diese gar nicht benötigt wird. Das kostet Speicher- und Prozessorleistung. Zwar können Sie auch einfache Grafiken auf die Bühne legen. Diese können aber nicht mit ActionScript gesteuert oder in die hierarchische Baumstruktur eingebunden werden, weshalb Sie auch nur selten zweckmäßig sind. ActionScript 3 verfügt über mehrere Typen von Anzeigeobjekten, die auf bestimmte Zwecke zugeschnitten sind und nicht immer die komplette Funktionalität eines Movieclips mit sich »herumschleppen« müssen.

Um nun in ActionScript 3 ein Anzeigeobjekt auf die Bühne zu holen, wird die Funktion `addChild()` verwendet. Als Parameter erwartet sie das anzufügende Objekt, das natürlich existieren muss. Der folgende Code erstellt einen neuen Movieclip, zeichnet ein Rechteck hinein und holt ihn dann auf die Bühne.

Video-Training

Wie Sie richtig mit Movieclips und der Anzeigeliste umgehen, können Sie in der Video-Lektion 3.3 auf der Buch-DVD einmal »live« miterleben.

```
1: var mc:MovieClip = new MovieClip();
2: mc.graphics.lineStyle(2,0x000000,100);
3: mc.graphics.drawRect(0,0,100,50);
4: this.addChild(mc);
```

Wie Sie sehen, wird zunächst mit `new()` eine Instanz von `MovieClip` erstellt, bevor sie mit `addChild()` auf die Bühne geholt wird. Das ist neu. In ActionScript 2 konnte man Movieclips nur direkt auf die Bühne holen, etwa mit `attachMovie()`. Voraussetzung war, dass die Movieclips einen über die Bibliothek vergebenen Verknüpfungsnamen (Bezeichner) hatten. Das Erzeugen von Instanzen mit `new()` war somit nutzlos, da diese sowieso niemals im Flash-Film erschienen wären. Nur in der Bibliothek verknüpfte Movieclips konnten auf die Bühne geholt werden.

Durch die neue Möglichkeit, Anzeigeobjekte zu erstellen, die nicht sofort auf der Bühne erscheinen müssen, ist eine wesentlich höhere Dynamik in Flash-Projekten möglich. Selbst wenn Sie Symbole aus der Bibliothek auf die Bühne holen, wird diese Syntax jetzt beibehalten. Anstelle eines Bezeichners geben Sie jetzt eine Klasse an. Diese wird dann ebenfalls mit `new()` instantiiert. Wenn Sie beispielsweise einen Movieclip in der Bibliothek haben und diesem als Klassennamen »mc« zugewiesen haben, können Sie ihn folgendermaßen auf die Bühne holen:

```
var mc_instanz:mc = new mc();
this.addChild(mc_instanz);
```

Neue Tiefenverwaltung

Die Tiefenverwaltung wurde in ActionScript 3 stark verbessert. Zunächst einmal müssen Sie die Tiefe nicht mehr explizit angeben. Sie ergibt sich ganz einfach aus der Reihenfolge, in der Sie die Objekte mit `addChild()` auf die Bühne geholt haben. Sie können also nicht mehr versehentlich Objekte löschen. Aber das ist erst der Anfang: Sie können auch die Tiefe vorhandener Objekte verändern, indem Sie die Methode `setChildIndex()` verwenden. Tun Sie das, passen sich die räumlichen Tiefen aller anderen Objekte automatisch an. Sie können sich das wie eine Reihe von Kugeln in einer offenen Röhre vorstellen. Entfernen Sie eine Kugel und fügen sie an anderer Stelle wieder ein, dann rutschen alle anderen Kugeln nach, so dass keine Lücken entstehen. Das erspart Ihnen viel Arbeit.

3D-Raum

Die räumliche Anordnung wird übrigens auch als »z-Sortierung« bezeichnet. Näheres dazu können Sie im Exkurs »3D-Programmierung in Flash CS3« auf Seite 184 lesen.

Partikel-Effekt

Bildpartikel animieren

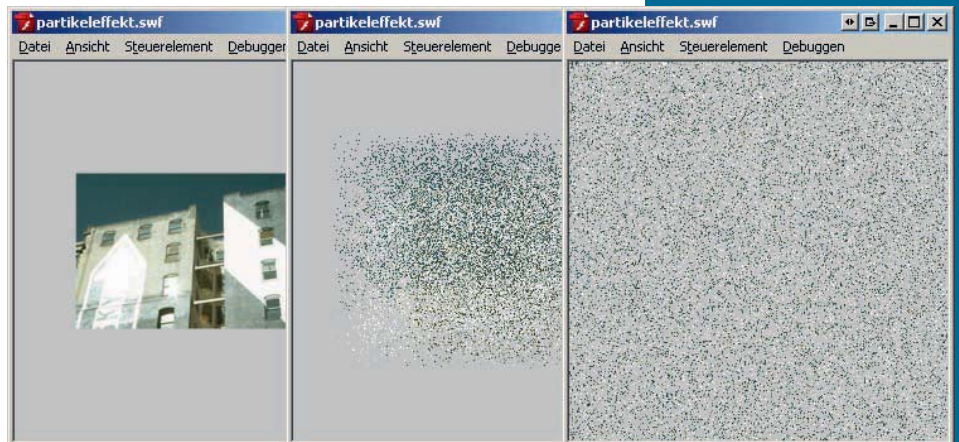
Mit dem Release von Flash Player 9 und Flash CS3 wurde die Flash-Gemeinde mit einem Performance-Geschenk bedacht. Bitmap-Operationen laufen nun mit einem Faktor von bis zu 10 schneller. Dies erlaubt es, jedes Pixel eines Quellbildes einzeln zu bewegen und zu animieren.

Zielsetzungen:

- BitmapData-Objekte anwenden
- Mit der setPixel-Methode arbeiten
- Mit Getter- und Setter-Methoden arbeiten

[Ergebnis: 11_PARTIKEL-EFFEKT • PARTIKELEFFEKT.SWF]

```
package
{
    import flash.display.MovieClip;
    import flash.display.BitmapData;
    import flash.display.Bitmap;
    import flash.events.MouseEvent;
    import flash.events.Event;
    public class Main extends MovieClip
    {
        private var original_bd:BitmapData;
        private var output_bd:BitmapData;
        private var particles:Array = new Array();
        private var output_bmp:Bitmap;
        private var particleMode:String;
        public function Main()
        {
            var num_x:Number = Math.floor(img.width);
            var num_y:Number = Math.floor(img.height);
            original_bd = new BitmapData(num_x, num_y, true, 0);
            original_bd.draw(img);
            output_bd = new BitmapData(300, 300, true, 0);
            output_bmp = new Bitmap();
        }
    }
}
```



Schritt 1 Dokument erstellen

Erstellen Sie ein neues AS3-Flash-Dokument, und öffnen Sie über **Strg**/**⌘** + **J** die Dokumenteigenschaften. Stellen Sie die Größe 300x300 Pixel und eine BILDRATE von 25 bps ein.



Abbildung 1 ►
Bereiten Sie das Flash-Dokument in den Dokumenteigenschaften vor.

Schritt 2 Quellbild erstellen

Erstellen Sie nun einen Movieclip, der als Quellbild für die Partikel dienen wird. Der Movieclip kann jedes visuelle Objekt erhalten. In diesem Beispiel nutzen wir ein Foto. Achten Sie beim Konvertieren des Fotos in einen Movieclip darauf, dass der Registrierungspunkt **1** Ihres Movieclips oben links ist.

Abbildung 2 ►
Konvertieren Sie das Bild in einen Movieclip mit dem Namen »img«.

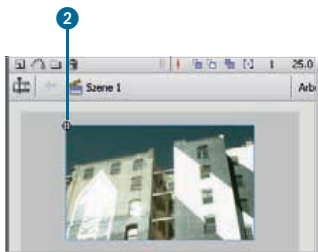
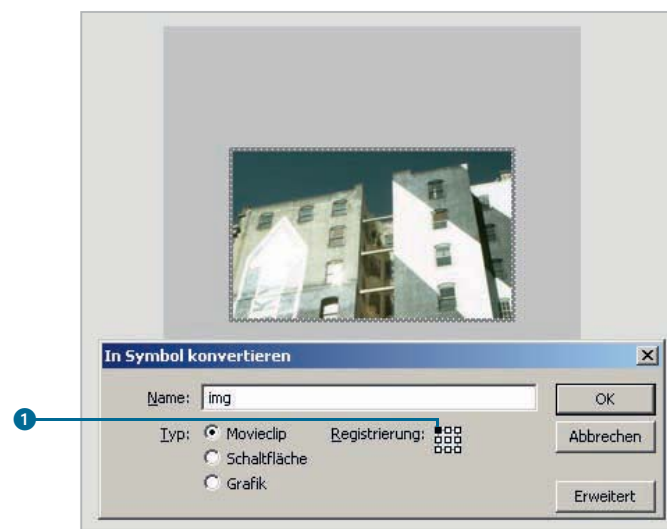
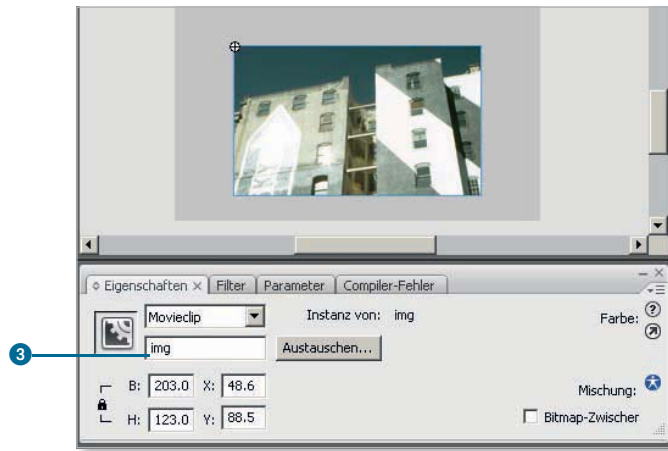


Abbildung 3
Der Registrierungspunkt **2** liegt in der linken oberen Ecke.



Geben Sie diesem Movieclip im Eigenschaftsfenster den Instanznamen »img« **3**.



Schritt 3

Quellmovieclip benennen

◀ **Abbildung 4**
Geben Sie dem Movieclip den Instanznamen »img«.

Erstellen Sie nun eine Dokumentklasse. Wählen Sie dafür den Menüpunkt **DATEI • NEU • ACTIONSCRIPT-DATEI**. Fügen Sie der Klasse folgende Zeilen hinzu. Es ist das Grundgerüst, das Sie später erweitern werden:

```
1: package
2: {
3:     import flash.display.MovieClip;
4:     import flash.display.BitmapData;
5:     import flash.display.Bitmap;
6:     import flash.events.MouseEvent;
7:     import flash.events.Event;
8:     public class Main extends MovieClip
9:     {
10:         private var original_bd:BitmapData;
11:         private var output_bd:BitmapData;
12:         private var particles:Array = new Array();
13:         private var output_bmp:Bitmap;
14:         private var particleMode:String;
15:         public function Main()
16:         {
17:         }
18:     }
19: }
```

In den Zeilen 3 bis 7 importieren Sie die Klassen, die für die Erstellung des Partikel-Effekts benötigt werden.

Schritt 4

Dokumentklasse erstellen

In dem Konstruktor, ab Zeile 10, werden die Variablen deklariert, die Sie für den Effekt benötigen. Mehr dazu später.

Speichern Sie diese Datei unter dem Namen `MAIN.AS` in demselben Verzeichnis wie die `fla`-Datei.

Schritt 5 Dokumentklasse verknüpfen

Verknüpfen Sie Ihre Dokumentklasse nun mit der `fla`-Datei, indem Sie im Feld `DOKUMENTKLASSE` im Eigenschaftfenster »Main« eingeben.



▲ Abbildung 5

Im Eigenschaftfenster können Sie, wenn kein Element auf der Bühne ausgewählt ist, die Dokumentklasse verknüpfen.

Schritt 6 Partikel-Klasse erstellen

Erstellen Sie über `DATEI • NEU • ACTIONSCRIPT-DATEI` eine neue Klasse. Speichern Sie diese unter dem Namen `PARTICLE.AS` in demselben Verzeichnis wie die `fla`-Datei.

Fügen Sie der Partikel-Klasse folgenden Code hinzu:

```
1: package{
2:     import fl.motion.easing.Cubic;
3:     public class Particle
4:     {
5:         private var startX:Number;
6:         private var startY:Number;
7:         private var beginX:Number;
8:         private var beginY:Number;
9:         private var changeX:Number;
10:        private var changeY:Number;
11:        private var currentX:Number;
12:        private var currentY:Number;
13:        private var targetX:Number;
14:        private var targetY:Number;
15:        private var cColor:int;
16:        private var t:int;
17:        private var d:int = 40;
```

```

18:     public function Particle(x:Number, y:Number, c:Number)
19:     {
20:         startX = x;
21:         startY = y;
22:         cColor = c;
23:         currentX = startX;
24:         currentY = startY;
25:     }
26: }
27: }

```

In Zeile 2 importieren Sie die Ease-Klasse, um die Partikel in einer organischen Bewegung zu animieren.

Ab Zeile 3 werden die Variablen deklariert, die später für die Partikel-Klasse benötigt werden.

`startX` und `startY` in den Zeilen 5 und 6 bezeichnen die Ausgangsposition des Partikels, also die Position, die der Partikel hat, wenn er noch unbewegt ist.

In den folgenden Zeilen (9–17) werden nun die Variablen für die Ease-Funktion, also für die Bewegung deklariert. Dazu mehr im nächsten Schritt.

In den Zeilen 11 und 12 werden mit `currentX` und `currentY` die Variablen deklariert, die später die aktuelle x- und y-Position des Partikels speichern.

Die Variable `cColor` in Zeile 15 speichert die Farbe des Partikels.

Dem Konstruktor in Zeile 18 werden die Parameter `x`, `y` und `c` übergeben. Es handelt sich hierbei um die Ausgangsdaten des Partikels, die in den folgenden Zeilen in den entsprechenden Variablen gespeichert werden.

Fügen Sie Ihrem Quellcode nun unter dem Konstruktor folgende Zeilen hinzu:

```

1: public function move():void
2: {
3:     if(++t >= d) return;
4:     currentX = Cubic.easeInOut(t, beginX, changeX, d);
5:     currentY = Cubic.easeInOut(t, beginY, changeY, d);
6: }

```

Ease-Funktionen

Ausführliche Informationen über die Ease-Klasse finden Sie in der Flash-Hilfe. Außerdem ist die Website von Robert Penner zu empfehlen. Hier finden Sie auch eine Demoseite, auf der die einzelnen Ease-Funktionen sehr gut visualisiert werden:

- <http://www.robertpenner.com/>
- http://www.robertpenner.com/easing/easing_demo.html

Schritt 7

Bewegungsfunktion der Partikel-Klasse

Die Methode `move()` wird später aus `MAIN.AS` heraus aufgerufen. Sie berechnet die neue Position des Partikels.

Die Ease-Methoden sind Hilfsfunktionen, die Bewegungen organisch erscheinen lassen. Sie berechnen aus dem Start- und Endwert die entsprechenden Zwischenschritte. Die Ease-Methoden benötigen folgende Parameter:

- ▶ `timer`: ein fortlaufender Wert, der in jedem Aufruf um 1 hochgezählt wird. Dient zur Bestimmung, welcher Schritt der Animation berechnet werden soll.
- ▶ `begin`: Ausgangswert
- ▶ `change`: der Wert, um den der Ausgangswert verändert werden soll
- ▶ `duration`: Dauer der Berechnung. Der Wert legt fest, wie viele Berechnungsschritte durchgeführt werden sollen.

In Zeile 3 wird zunächst die Variable `t` mit eins addiert. Anschließend wird überprüft, ob `t` die Variable `d` schon überschritten hat. Wenn dies der Fall ist, wird die Funktion mit `return`; abgebrochen.

In Zeile 4 wird die aktuelle x-Position berechnet. Dazu wird als erster Parameter `t` übergeben, der in der vorigen Zeile in jedem Durchgang um 1 erhöht wird (`t++`).

Als zweiter Parameter (`beginX`) wird der Wert übergeben, der die Anfangsposition der Animation angibt.

Parameter 3 (`changeX`) ist der Wert, um den der Ausgangswert geändert werden soll.

Als vierten und letzten Parameter (`d`) übergeben Sie schließlich die Anzahl der Berechnungsschritte.

Dasselbe geschieht für die y-Position in Zeile 5.

Schritt 8

Startmethode der Partikel-Klasse

Nun fügen Sie Ihrer Partikel-Klasse die Methoden hinzu, die die jeweiligen Endwerte, also die Zielposition des Partikels, setzen. Auch diese Funktionen werden später aus `MAIN.AS` heraus aufgerufen.

Ergänzen Sie Ihr Skript unter der `move`-Methode um folgenden Code:

```
1: public function setModeOut():void
2: {
3:     t = 0;
4:     beginX = currentX;
5:     beginY = currentY;
6:     targetX = Math.random()*300;
```

```

7:     targetY = Math.random()*300;
8:     changeX = targetX - beginX;
9:     changeY = targetY - beginY;
10: }
11: public function setModeIn():void
12: {
13:     t = 0;
14:     beginX = currentX;
15:     beginY = currentY;
16:     targetX = startX;
17:     targetY = startY;
18:     changeX = startX - currentX;
19:     changeY = startY - currentY;
20: }

```

Die Partikel werden zwei Richtungen haben: eine von der Ausgangsposition weg und eine zur Ausgangsposition zurück.

Die Methode `setModeOut()` in Zeile 1 setzt die Werte für die Animation, bei der die Partikel aus dem Bild herausfliegen.

In Zeile 3 wird die fortlaufende Variable `t` auf 0 zurückgesetzt.

`beginX` und `beginY` werden in den Zeilen 4 und 5 auf den Wert der derzeitigen x- und y-Position gesetzt.

In den Zeilen 6 und 7 werden die Zielpositionen festgelegt. Dabei werden `targetX` und `targetY` jeweils mit Hilfe der `Math.random`-Funktion auf einen Zufallswert zwischen 0 und 300 (Bühnenhöhe und -breite) gesetzt.

Die Änderungswerte `changeX` und `changeY` werden in den Zeilen 8 und 9 jeweils aus der Differenz zwischen Start- und Endwert errechnet.

Die `setModeIn`-Methode in Zeile 11 ist für die Zurück-Bewegung zur Ausgangsposition zuständig und tut im Grunde genau dasselbe. Bis auf den Unterschied, dass in den Zeilen 16 und 17 die Zielwerte für die X- und Y-Position, `targetX` und `targetY`, wieder auf die Werte der Ausgangsposition, `startX` und `startY`, zurückgesetzt werden.

Sie werden der Partikel-Klasse nun Getter-Methoden für die Variablen hinzufügen, die später aus `MAIN.AS` ausgelesen werden. Setter-Methoden benötigen Sie hier nicht, weil die Variablen aus `MAIN.AS` lediglich ausgelesen, nicht aber gesetzt werden.

Ergänzen Sie Ihr Skript unter dem in Schritt 8 hinzugefügten Code wie folgt:

Zufallszahlen

`Math.random()` gibt eine Pseudozufallszahl zwischen 0 und 1 zurück. `random(n)` gibt (nur in AS2) eine Zufallszahl zwischen 0 und n zurück.

Schritt 9

Getter-Methoden für die Partikel-Klasse

Getter- und Setter-Methoden

Es ist gute Praxis, Variablen nicht direkt auszulesen oder zu setzen, sondern dies über Getter- und Setter-Methoden zu tun. Dies hat den Vorteil, dass vor dem eigentlichen Setzen des Wertes z. B. überprüft werden kann, ob der neue Wert überhaupt gültig ist. Ausführliche Informationen zur Verwendung von Getter- und Setter-Methoden erhalten Sie in der Flash-Hilfe.

```
public function get x():Number
{
    return currentX;
}
public function get y():Number
{
    return currentY;
}
public function get color():int
{
    return cColor;
}
```

Wenn aus MAIN.AS jetzt die Variablen `x`, `y` oder `color` aufgerufen werden, gibt die Partikel-Klasse den entsprechenden Wert zurück.

Schritt 10

Main.as erweitern

Die Partikel-Klasse PARTICLE.AS ist nun fertiggestellt. Für jedes Pixel des Quellbildes wird eine Instanz der Partikel-Klasse erstellt und x-, y-Position sowie Farbe des Pixels werden übergeben.

Nun geht es mit der Datei MAIN.AS weiter. Hier werden Sie die eigentlichen Partikel erstellen.

Schritt 11

Anzahl der Partikel auslesen

Fügen Sie folgenden Code in die `Main`-Funktion (Zeile 17) der MAIN.AS ein:

```
var num_x:Number = Math.floor(img.width);
var num_y:Number = Math.floor(img.height);
```

In diesen beiden Zeilen speichern Sie jeweils die Anzahl der Partikel horizontal und vertikal, indem Sie die Breite und Höhe des auf der Bühne befindlichen `img-Movieclips` auslesen. Dieser Wert wird mit der Methode `Math.floor()` abgerundet.

Schritt 12

Bitmaps erstellen

Ergänzen Sie Ihr Skript direkt im Anschluss an die beiden Zeilen aus Schritt 11 um folgende Zeilen:

```
1: original_bd = new BitmapData(num_x, num_y, true, 0);
2: original_bd.draw(img);
```

```

3: output_bd = new BitmapData(300, 300, true, 0);
4: output_bmp = new Bitmap();
5: output_bmp.bitmapData = output_bd;
6: addChild(output_bmp);

```

Um die Bitmapdaten von `img` auslesen zu können, erstellen Sie in Zeile 1 ein `BitmapData`-Objekt mit der Breite und Höhe von `img` (`num_x`, `num_y`). Das Objekt soll Transparenzen unterstützen (`true`) und zunächst keine Farbe enthalten (`0`).

In Zeile 2 zeichnen Sie anschließend ein Abbild von `img` in `original_bd`.

In Zeile 3 erstellen Sie ein weiteres `BitmapData`-Objekt `output_bd`, das später die Ausgabedaten enthalten wird. Dieses Objekt hat die Größe der Bühne und ist zunächst ebenfalls leer.

Die Darstellung der Ausgabedaten übernimmt ein weiteres `Bitmap`-Objekt. Dieses erstellen Sie in Zeile 4. In der nächsten Zeile legen Sie fest, dass `output_bmp` zur Darstellung die Bitmapdaten von `output_bd` nehmen soll. Mit `addChild()` in Zeile 5 wird die Bitmap auf der Bühne angezeigt.

Nun erstellen Sie die Partikel. Fügen Sie, wieder direkt im Anschluss an die oberen Zeilen, dem Skript folgenden Code hinzu:

```

1: particles = new Array();
2: for(var i:int = 0; i < num_x; i++)
3: {
4:     for(var j:int = 0; j < num_y; j++)
5:     {
6:         var p:Particle = new Particle(img.x + i, img.y + j,
            original_bd.getPixel32(i, j));
7:         if(p.color != 0)
8:         {
9:             particles.push(p);
10:        }
11:    }
12: }

```

Zunächst erstellen Sie in Zeile 1 ein Array, in dem alle erstellten Partikel gespeichert werden.

Jetzt wird mit Hilfe von zwei `for`-Schleifen (Zeilen 2 bis 5) das Quellbild (`img`) horizontal und vertikal ausgelesen.

Rückgabewerte

Funktionen oder Methoden dienen oft der Berechnung eines Wertes. Der Methode werden Werte als Parameter übergeben, die die Methode dann zur Berechnung nutzt. Anschließend gibt die Funktion den fertig berechneten Wert als Rückgabewert zurück.

Die Rückgabewerte der Flash-eigenen Funktionen können Sie der Flash-Hilfe entnehmen.

Schritt 13

Partikel erstellen

Video-Training

Arrays sind ein mächtiges Mittel, um Daten zu strukturieren. Die Video-Lektion 2.3 auf der Buch-DVD gibt Ihnen einen Einblick in die vielfältigen Möglichkeiten.

getPixel32

Im Unterschied zur `getPixel`-Methode gibt Ihnen die `getPixel32`-Methode auch den Alphawert des ausgelesenen Pixels zurück (ARGB).

Nun endlich, in Zeile 6, erstellen Sie Ihre Partikel. Dabei übergeben Sie dem `Particle`-Objekt die x-Position von `img` plus der x-Position, die aktuell ausgelesen wird. Dasselbe passiert für den y-Wert.

Außerdem lesen Sie mit Hilfe der `getPixel32`-Funktion den Farbwert des aktuellen Pixels aus.

Bevor das Pixel nun in Zeile 9 dem Array `particles` hinzugefügt wird, überprüfen Sie in Zeile 7, ob das Pixel transparent, also nicht sichtbar ist. Sie können statt des Fotos auch ein Textfeld verwenden, das viele transparente Bildteile zwischen den Buchstaben hat – die beim Zeichnen (siehe Zeile 2 in Schritt 12) aber trotzdem in `original_bd` gespeichert werden. Wenn das Pixel also transparent ist, benötigen Sie diesen Partikel nicht, er ist sowieso nicht sichtbar. So sparen Sie in Ihrem Skript Performance; denn je weniger Partikel, desto weniger wird der Prozessor des Rechners beansprucht.

Schritt 14

Mouse-Event integrieren

Nun haben Sie alle Pixel Ihres Quellbildes als `Particle`-Objekt in einem Array gespeichert. Noch fliegen die Partikel aber nicht. Dies soll bei einem Mausklick des Users passieren.

Ergänzen Sie die `Main`-Funktion ganz unten, außerhalb der beiden `for`-Schleifen um folgende Zeilen:

```
stage.addEventListener(MouseEvent.CLICK, onStageMouseDown);
particleMode = "none";
```

Die erste Zeile bewirkt, dass bei einem Mausklick des Users auf der Bühne die (noch anzulegende) Funktion `onStageMouseDown()` aufgerufen wird.

Außerdem setzen Sie in der zweiten Zeile die Variable `particleMode` auf `"none"`. Diese Variable speichert, in welcher Bewegung sich die Partikel gerade befinden.

Schritt 15

Mouse-Event anlegen

Jetzt werden Sie die Funktion anlegen, die auf den Mausklick des Users reagieren soll. Fügen Sie folgende Zeilen unterhalb der nun fertig gestellten `Main`-Funktion in Ihrem Quellcode hinzu:

```
1: private function onStageMouseDown(evt:MouseEvent):void
2: {
3:     var i:String, p:Particle;
4:     if(particleMode == "none")
```

```

5:     {
6:         img.visible = false;
7:         addEventListener(Event.ENTER_FRAME, onEnterFrame);
8:         for(i in particles)
9:         {
10:            p = particles[i];
11:            p.setModeOut();
12:        }
13:        particleMode = "out";
14:    }
15: }

```

Die Funktion `onStageMouseDown()` wird immer aufgerufen, wenn der User auf die Bühne klickt.

In Zeile 3 initialisieren Sie zunächst die Variablen, die in dieser Funktion benötigt werden.

Wenn sich die Partikel noch im Wartemodus (`particleMode == "none"`) befinden, machen sie das Quellbild in Zeile 6 zunächst unsichtbar. Von nun an werden die Partikel die Zeichnung des Bildes übernehmen.

In Zeile 7 melden Sie sich beim `ENTER_FRAME`-Event `onEnterFrame` an. Ab jetzt wird in jedem Frame die (noch zu erstellende) Funktion `onEnterFrame()` aufgerufen.

Nun durchläuft das Skript in Zeile 8 das `particles`-Array. Jedem Partikel wird mit der Funktion `setModeOut()` mitgeteilt, wie es sich von nun an zu bewegen hat.

In Zeile 13 schließlich setzen Sie den Modus auf `"out"`. So weiß das Skript, was beim nächsten Mausklick zu tun ist.

Jetzt werden Sie die eigentliche Bewegungsfunktion für die Partikel hinzufügen. Ergänzen Sie Ihr Skript um folgende Zeilen und fügen Sie unter der Funktion `onStageMouseDown()` eine weitere Funktion ein:

```

1: private function onEnterFrame(evt:Event):void
2: {
3:     output_bd.fillRect(output_bd.rect, 0x00000000);
4:     output_bd.lock();
5:     for(var i in particles)
6:     {
7:         var p:Particle = particles[i];
8:         p.move();

```

Schritt 16

Bewegungsskript

```

9:         output_bd.setPixel32(p.x, p.y, p.color);
10:     }
11:     output_bd.unlock();
12: }

```

Wie vorher schon beschrieben, wird die `onEnterFrame`-Funktion in jedem Frame ausgeführt.

Mit `fillRect()` in Zeile 3 wird das `BitmapData`-Objekt, das für die Ausgabe der Partikel zuständig ist, mit einer transparenten Fläche (`0x00000000`) übermalt. So löschen Sie alle alten Partikel. (Sonst würden die neuen Partikel im nächsten Schritt über die alten gezeichnet.)

Anschließend wird das Array in Zeile 5 durchlaufen. In Zeile 7 schließlich wird die `move`-Funktion der in dem `particles`-Array gespeicherten Partikel angestoßen.

Nun schreiben Sie die neuen Daten (`x`, `y`, `color`) mit `setPixel32()` in das Ausgabe `BitmapData`-Objekt `output_bd`.

Zu `lock()` und `unlock()` in den Zeilen 4 und 11: Standardmäßig löst die `setPixel32`-Methode ein sofortiges Aktualisieren von `output_bd` (und damit auch von `output_bmp`) aus. Wir möchten aber, dass die Schleife erst komplett durchläuft und die Ausgabe dann aktualisiert wird – dies spart Performance. Dieses Verhalten erreichen Sie, indem Sie vor der Schleife `lock()` und nach der Schleife `unlock()` aufrufen.

Schritt 17

Film testen

Nun können Sie Ihr Skript zum ersten Mal testen. Wenn Sie nun auf die Bühne klicken, strömen die Partikel in alle Richtungen auseinander.

Schritt 18

Weitere Partikelbewegungen hinzufügen

Zwischenergebnis:
12_PARTIKEL-EFFEKT •
MAIN_STEP17.AS



```

1: else if(particleMode == "out")
2: {
3:     for(i in particles)
4:     {
5:         p = particles[i];
6:         p.setModeIn();
7:     }
8:     particleMode = "in";
9: }

```



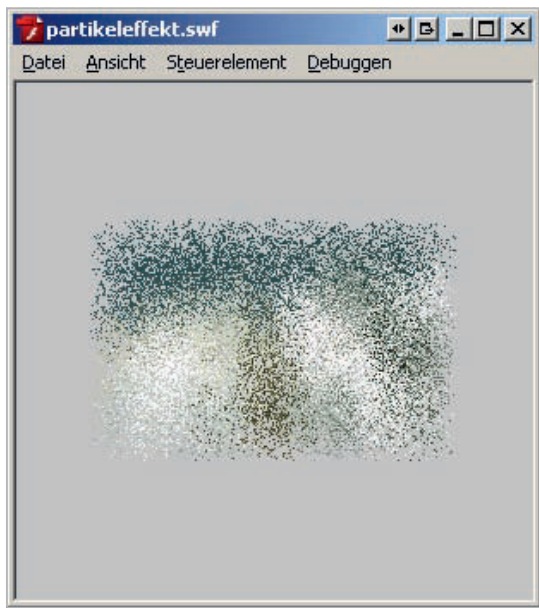
```

10: else if(particleMode == "in")
11: {
12:     for(i in particles)
13:     {
14:         p = particles[i];
15:         p.setModeOut();
16:     }
17:     particleMode = "out";
18: }

```

Wenn der aktuelle Modus also "out" ist (Zeile 1), wird wieder das Partikel-Array durchlaufen. Den Partikeln wird mit der Methode `setModeIn()` mitgeteilt, dass sie sich nun wieder zu ihrem Ursprungspunkt zurückbewegen sollen.

Anschließend wird der Modus in Zeile 8 auf "in" gesetzt. Wenn der Modus nun "in" ist, wird eine weitere Abfrage benötigt (Zeile 10). Sie unterscheidet sich nur insofern von der "none"-Abfrage, dass eine Anmeldung bei `ENTER_FRAME` und das Unsichtbarmachen des Quellbildes `img` nicht erforderlich sind.



◀ **Abbildung 6**
 Fliegende Partikel – ganz
 einfach per Mausklick

Flüssigkeit simulieren

Wasserspritzer mit der Maus erzeugen

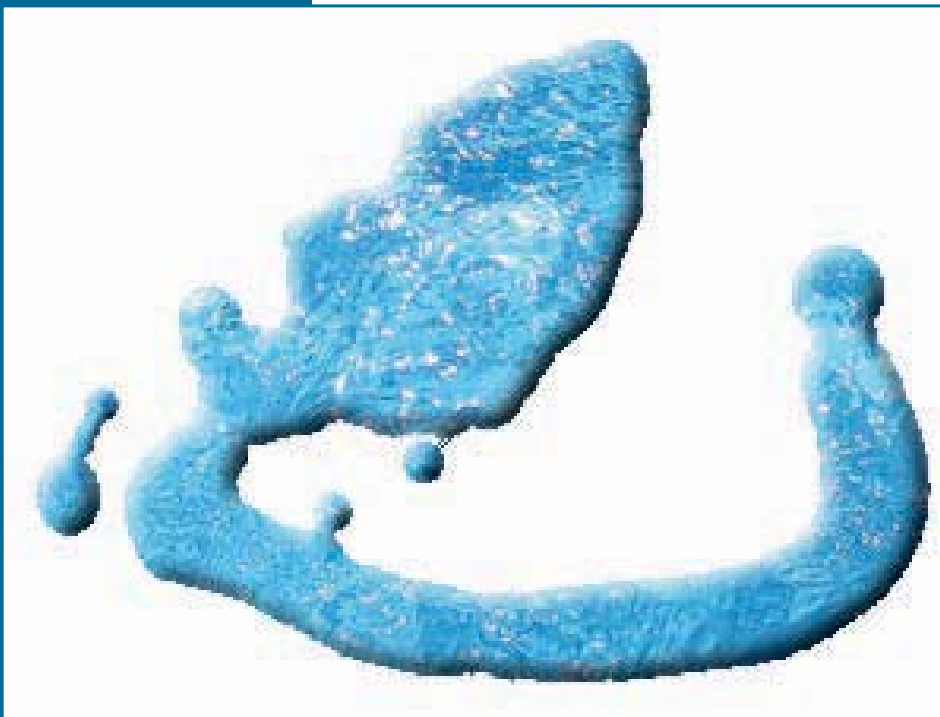
```
package
{
    import flash.display.MovieClip;
    import flash.display.BitmapData;
    import flash.events.*;
    import flash.display.Bitmap;
    import flash.geom.ColorTransform;
    import flash.filters.BevelFilter;
    import flash.geom.Point;
    import flash.geom.Matrix;
    public class Main extends MovieClip
    {
        private var original_water_bd:BitmapData;
        private var output_bd:BitmapData;
        private var grey_bd:BitmapData;
```

Teilweise sind die Bitmap-Funktionen von Flash schon etwas verwirrend. Wenn Sie im Internet mal nach der threshold-Funktion suchen, werden Sie die seltsamsten Effekte finden. Richtig angewandt, können Sie mit der Funktion aber auch wirklich beeindruckende Effekte erzielen. In diesem Workshop nutzen Sie die threshold-Funktion, um einen Wassereffekt zu erzeugen.

Zielsetzungen:

- BitmapData-Objekt anwenden
- threshold-Funktion einsetzen
- ColorTransform-Funktion nutzen
- BevelFilter anwenden

[Ergebnis: 12_FLUESSIGKEIT •
EVAPORATINGDRIP.SWF]



Erstellen Sie ein neues AS3-Flash-Dokument. Stellen Sie in den Dokumenteigenschaften (**Strg**/**⌘**+**J**) die Grösse 600x300 Pixel und eine BILDRATE von 25 bps ein.



Schritt 1

Dokument erstellen

◀ Abbildung 1

Dokumenteigenschaften festlegen

Erstellen Sie einen Movieclip, der als Wasser-Bild für die spätere Animation dient. In diesem Beispiel wird ein Bild (WATER.JPG) genutzt, das einen Swimmingpool zeigt. Sie können aber auch ein anderes Bild nehmen. Wichtig ist, dass dieses Bild eine Breite und Höhe von je 300 Pixel hat.

Achten Sie außerdem darauf, dass der Registrierungspunkt des Movieclips oben links ist.



Schritt 2

Wasserbild erstellen

Movieclips erstellen

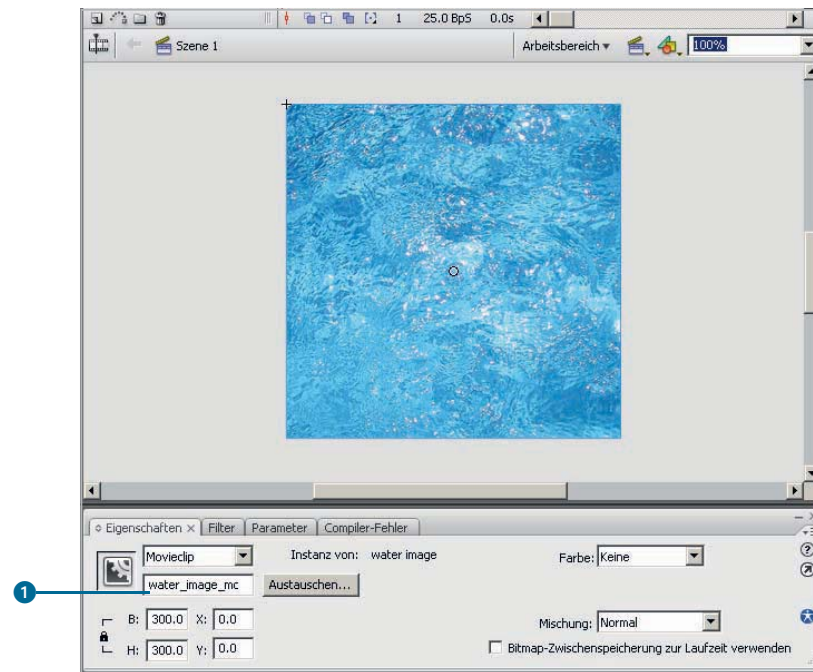
Eine kurze Anleitung, wie Sie ein importiertes Bild in einen Movieclip umwandeln, finden Sie im Workshop »Wellen simulieren« in Schritt 3.

◀ Abbildung 2

Konvertieren Sie das Wasser-Bild in einen Movieclip.

Ziehen Sie diesen Movieclip nun auf die Bühne, und geben Sie ihm im Eigenschaftenfenster den Instanznamen »water_image_mc« **1**.

Abbildung 3 ►
Geben Sie dem Wasser-Movieclip den Instanznamen »water_image_mc«.

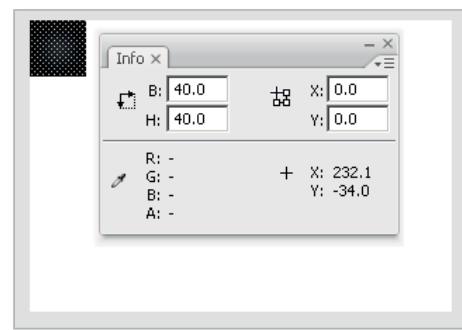


Schritt 3 Werkzeugspitze erstellen

Nun erstellen Sie die Werkzeugspitze, mit der der User später die Flüssigkeit auf dem Screen erzeugen kann. Dazu wählen Sie in der Werkzeugpalette das Rechteckwerkzeug .

Ziehen Sie ein Quadrat der Größe 40 x 40 Pixel auf. Die Randfarbe ist transparent; die Füllfarbe werden Sie im nächsten Schritt einstellen.

Abbildung 4 ►
Die Größe des Rechtecks können Sie in der INFO-Palette eingeben.

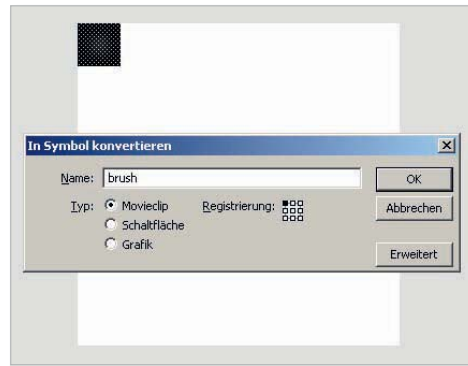
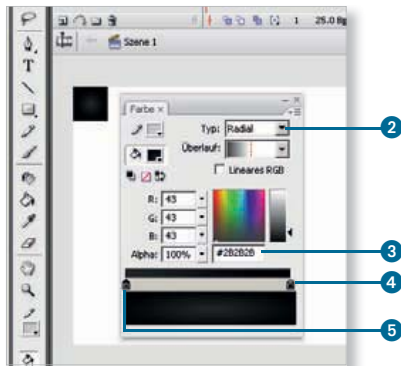


Öffnen Sie nun die Palette FARBE, und wählen Sie unter TYP einen radialen Verlauf . Sollte die Palette bei Ihnen nicht zu sehen sein, wählen Sie im Menü FENSTER • FARBE.

Klicken Sie nun die beiden Farbeimer links  und rechts  an, um die Farben anzupassen: Der Farbwert auf der linken Seite ist #2B2B2B, auf

der rechten Seite Schwarz (#000000). Sie können die Werte einfach in das Feld ③ eingeben.

Wandeln Sie das Quadrat nun über MODIFIZIEREN • IN SYMBOL KONVERTIEREN... oder [F8] in einen Movieclip mit dem Namen »brush« um.



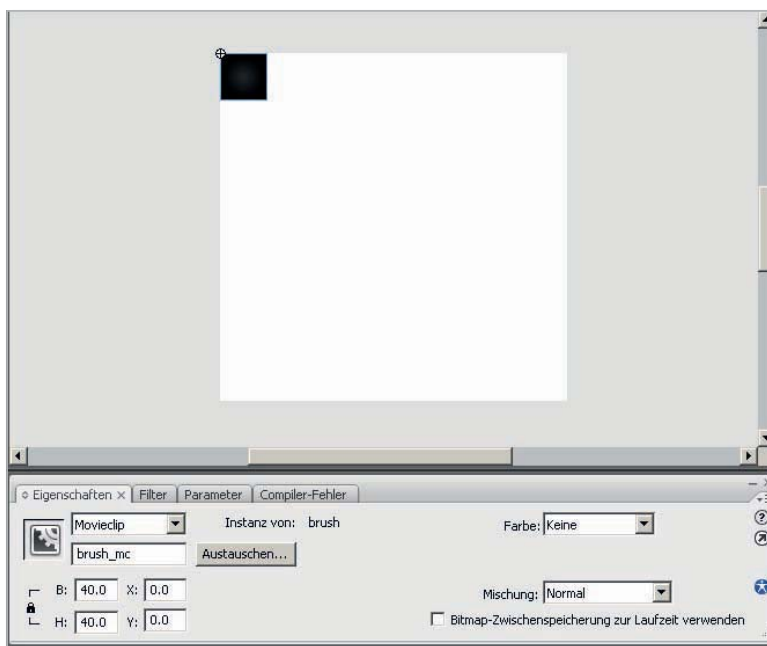
◀◀ **Abbildung 5**

Im FARBE-Bedienfeld stellen Sie den gewünschten Farbverlauf ein.

◀ **Abbildung 6**

Konvertieren Sie die Fläche in einen Movieclip.

Nach der Bestätigung mit OK geben Sie dem Movieclip im Eigenschaftenfenster den Instanznamen »brush_mc«.



◀ **Abbildung 7**

Geben Sie dem Movieclip den Instanznamen »brush_mc«.

Erstellen Sie nun eine Dokumentklasse. Klicken Sie dafür auf DATEI • NEU • ACTIONSCRIPT-DATEI. Fügen Sie der Klasse folgende Zeilen hinzu. Es ist das Grundgerüst, das Sie später erweitern werden.

Schritt 4

Dokumentklasse erstellen

Zwischenergebnis:
12_FLUESSIGKEIT •
MAIN_STEP04.AS



```
1: package
2: {
3:     import flash.display.MovieClip;
4:     import flash.display.BitmapData;
5:     import flash.events.*;
6:     import flash.display.Bitmap;
7:     import flash.geom.ColorTransform;
8:     import flash.filters.BevelFilter;
9:     import flash.geom.Point;
10:    import flash.geom.Matrix;
11:    public class Main extends MovieClip
12:    {
13:        private var original_water_bd:BitmapData;
14:        private var output_bd:BitmapData;
15:        private var grey_bd:BitmapData;
16:        private var brush_bd:BitmapData;
17:        private var display_bmp:Bitmap;
18:        private var map_bmp:Bitmap;
19:        private var reducecolor:ColorTransform;
20:        private var bevel:BevelFilter;
21:        public function Main()
22:        {
23:        }
24:    }
25: }
```

In den Zeilen 3 bis 10 importieren Sie zunächst die Klassen, die für die Programmierung benötigt werden.

In den Zeilen 13 bis 20 deklarieren Sie alle benötigten Variablen.

Speichern Sie die Datei, und fügen Sie diese Klasse Ihrer fla-Datei als Dokumentklasse an. Klicken Sie dazu auf eine leere Stelle auf der Bühne und geben Sie »Main« im Eigenschaftfenster **1** ein.

Abbildung 8 ►

Die Dokumentklasse wird
im Eigenschaftfenster
definiert.



Schritt 5

Movieclips unsichtbar schalten

Fügen Sie folgenden Code in der Main-Funktion hinzu:

```
water_image_mc.visible = false;
brush_mc.visible = false;
```

Zunächst schalten Sie die Sichtbarkeit beider Movieclips auf der Bühne auf unsichtbar. Sie benötigen für den Brush zwar die Bitmapdaten des Bildes, der Movieclip selbst soll aber nicht sichtbar sein.

Ergänzen Sie Ihre Klasse MAIN.AS direkt unter den beiden Codezeilen aus Schritt 5 wie folgt:

```
original_water_bd = new BitmapData(300, 300, false, 0);  
original_water_bd.draw(water_image_mc);
```

In der ersten Zeile erstellen Sie ein neues BitmapData-Objekt, das eine Breite und Höhe von jeweils 300 Pixel, also dieselbe Größe wie der zuvor erstellte Movieclip `water_image_mc` hat. Außerdem soll das BitmapData-Objekt keine Transparenzen unterstützen (`false`) und zunächst keine Farbe enthalten (`0`).

Nun zeichnen Sie in der zweiten Zeile ein Abbild des Movieclips in das BitmapData-Objekt `original_water_bd`.

Nun erstellen Sie die Objekte, die für die Ausgabe des Effekts benötigt werden. Fügen Sie, wieder direkt unter den Anweisungen aus dem vorigen Schritt, folgende Zeilen hinzu:

```
1: output_bd = new BitmapData(300, 300, true, 0);  
2: display_bmp = new Bitmap();  
3: display_bmp.bitmapData = output_bd;  
4: addChild(display_bmp);
```

In Zeile 1 erstellen Sie ein BitmapData-Objekt, das wiederum die Größe 300x300 Pixel hat. Im Unterschied zu `water_image_mc` soll `output_bd` aber Transparenzen unterstützen (`true`), da im Ergebnis nur bestimmte Bildteile mit Farbe gefüllt werden und der Rest unsichtbar sein wird.

Um die Bitmapdaten nun ausgeben zu können, erstellen Sie in Zeile 2 ein Bitmap-Objekt. Dieses weisen Sie in Zeile 3 an, die Bitmapdaten von `output_bd` als Ausgabedaten zu nutzen. In Zeile 4 schließlich fügen Sie das Bitmap Objekt `display_bmp` der Bühne hinzu.



Zwischenergebnis:
12_FLUESSIGKEIT •
MAIN_STEP05.AS

Schritt 6

Wasserbilddaten aufnehmen



Zwischenergebnis:
12_FLUESSIGKEIT •
MAIN_STEP06.AS

Schritt 7

Ausgabe-Bitmap erstellen



Zwischenergebnis:
12_FLUESSIGKEIT •
MAIN_STEP07.AS

Schritt 8

Map-BitmapData erstellen

Zwischenergebnis:

12_FLUESSIGKEIT •
MAIN_STEP08.AS



Map

Mit Map ist in diesem Fall ein »Hilfs«-BitmapData-Objekt gemeint, das für die Berechnung des Ergebnisses benötigt wird. Auf einer Map werden Veränderungen der Bilddaten durchgeführt, die dann als Grundlage für das Ergebnis-Bitmap genutzt werden. Diese Map muss also nicht im Flash-Film angezeigt werden, da sie nur zur internen Berechnung dient.

Für den Flüssigkeitseffekt benötigen Sie ein BitmapData-Objekt, das später für die Bildteile des Ausgangsbildes, die sichtbar werden sollen, als »Schablone« dienen wird.

Ergänzen Sie Ihr Skript am Ende der `Main`-Funktion um folgende Zeilen:

```
grey_bd = new BitmapData(300, 300, true, 0xFF000000);  
map_bmp = new Bitmap();  
map_bmp.bitmapData = grey_bd;  
addChild(map_bmp);  
map_bmp.x = 300;
```

Zunächst erstellen Sie ein weiteres BitmapData-Objekt. In diesem Fall geben Sie ihm in der ersten Zeile den Namen `grey_bd`. Es hat wiederum die Größe 300x300 Pixel, soll Transparenzen unterstützen (`true`) und den Farbwert Schwarz (`0xFF000000`) haben.

Um Ihnen die Funktion dieses Map-BitmapData-Objekts deutlich zu machen, erstellen Sie noch ein Bitmap-Objekt zur Ausgabe. Es wird für den Effekt nicht benötigt, hilft aber, die Berechnungen zu verstehen.

Fügen Sie das Bitmap-Objekt `map_bmp` also der Bühne mit `addChild()` hinzu, und verschieben Sie es um 300 Pixel nach rechts, damit es nicht über dem Ausgabe-Bitmap-Objekt `display_bmp` liegt, das Sie im vorherigen Schritt erstellt haben.

Schritt 9

Die Werkzeugspitze verfügbar machen

Zwischenergebnis:

12_FLUESSIGKEIT •
MAIN_STEP09.AS



Der User soll später mit der Maus Flüssigkeit erzeugen können. Dafür benötigen Sie die zuvor erstellte Werkzeugspitze. Diese werden Sie nun dem Skript verfügbar machen. Erweitern Sie die `Main`-Funktion um folgende Zeilen:

```
brush_bd = new BitmapData(40, 40, true, 0);  
brush_bd.draw(brush_mc);
```

In der ersten Zeile erstellen Sie ein neues BitmapData-Objekt, das exakt die Größe des zuvor erstellten `brush_mc` hat und Transparenzen unterstützen soll (`true`).

In der zweiten Zeile zeichnen Sie nun ein Abbild von `brush_mc`.

Schritt 10

Events anmelden

Damit Ihr Skript registriert, dass die Maus bewegt wird, fügen Sie bitte folgende Zeilen hinzu:

```
stage.addEventListener(MouseEvent.CLICK, onStageMouseDown);
stage.addEventListener(MouseEvent.CLICK, onStageMouseUp);
```

Die Zeilen bewirken, dass immer wenn der User auf den Flash-Film klickt, die Funktionen `onStageMouseDown` und `onStageMouseUp` aufgerufen werden. Diese Funktionen werden im nächsten Schritt erstellt.

Ergänzen Sie Ihr Skript unter der `Main`-Funktion wie folgt:

```
1: private function onStageMouseDown(evt:MouseEvent):void
2: {
3:     stage.addEventListener(MouseEvent.CLICK,
        onStageMouseDown);
4: }
5: private function onStageMouseUp(evt:MouseEvent):void
6: {
7:     stage.removeEventListener(MouseEvent.CLICK,
        onStageMouseUp);
8: }
9: private function onStageMouseMove(evt:MouseEvent):void
10: {
11:     var ma = new Matrix();
12:     ma.tx = stage.mouseX - brush_mc.width/2;
13:     ma.ty = stage.mouseY - brush_mc.height/2;
14:     grey_bd.draw(brush_bd, ma, new ColorTransform(), "add");
15: }
```

`onStageMouseDown()` und `onStageMouseUp()` in den Zeilen 1 und 5 bewirken lediglich, dass sich das Skript bei dem `CLICK`-Event der Bühne (`stage`) an- bzw. abmeldet. Dadurch wird die Funktion `onStageMouseMove` ausgeführt, sobald der User seine Maus bewegt.

In der Funktion `onStageMouseMove()` ab Zeile 9 passiert Folgendes: `brush_bd` wird in Zeile 14 in `grey_bd` hineinkopiert. Damit der Mittelpunkt des radialen Verlaufs von `brush_bd` genau über der Mausposition kopiert wird, wird der Zeichenvorgang mit Hilfe einer Matrix (Zeile 11) verschoben. Und zwar in x- und y-Richtung jeweils um die Hälfte der Breite und Höhe von `brush_mc` (Zeilen 12 und 13).

Testen Sie nun Ihren Flash-Film über `STEUERUNG • FILM TESTEN` oder `[Strg]/` `[⌘]+[↩]`, und bewegen Sie die Maus mit gedrückter Maustaste auf dem



Zwischenergebnis:
12_FLUESSIGKEIT •
MAIN_STEP10.AS

Schritt 11

Maus-Funktionen hinzufügen



Zwischenergebnis:
12_FLUESSIGKEIT •
MAIN_STEP11.AS

Schritt 12

Film testen

Zwischenergebnis:
12_FLUESSIGKEIT •
MAIN_STEP12.AS



linken Teil (weiße Fläche) des Films. Sie werden sehen, dass auf der rechten Seite ② in der Ausgabe von `grey_bd` die weiße Farbe von `brush_bd` ① immer wieder hineinkopiert wird.

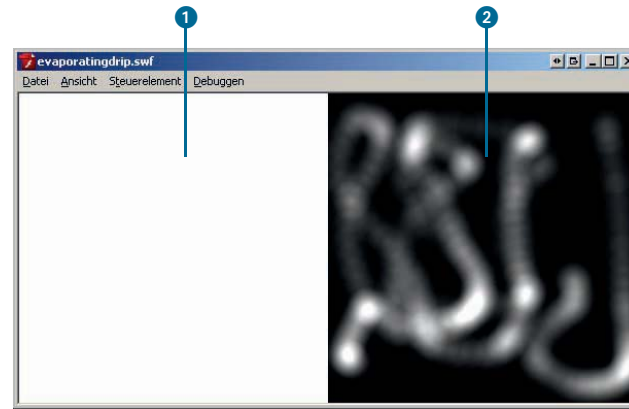


Abbildung 9 ►

Bei gedrückter Mausbewegung wird das Map-Bitmap-Data-Objekt verändert.

Schritt 13

Wasser-Bild sichtbar machen

Zwischenergebnis:
12_FLUESSIGKEIT •
MAIN_STEP13.AS



So weit, so gut: Die User-Interaktionen werden aufgenommen. Nun wollen wir das erstellte Wasser-Bild sichtbar machen. Dazu brauchen Sie zunächst ein weiteres Event. Fügen Sie folgenden Code direkt unter die beiden Event-Listener aus Schritt 10 in die `Main`-Funktion ein:

```
stage.addEventListener(Event.ENTER_FRAME, onStageEnterFrame);
```

Von nun an wird in jedem Frame die Funktion `onStageEnterFrame()` ausgeführt. Diese werden Sie nun erstellen. Ergänzen Sie Ihr Skript unter der letzten Funktion aus Schritt 11 um folgende Codezeilen:

```
private function onStageEnterFrame(evt:Event):void
{
    output_bd.copyPixels(original_water_bd, original_water_
        bd.rect, new Point(0, 0));
    output_bd.threshold(grey_bd, grey_bd.rect, new Point(0, 0),
        "==", 0x000000, 0, 0x00FFFF, false);
}
```

In der dritten Zeile werden die Bitmapdaten von `original_water_bd` in das Ausgabe-BitmapData-Objekt `output_bd` kopiert. Der zweite Parameter von `copyPixels()` gibt an, dass die komplette Fläche von `original_water_bd` kopiert werden soll, und zwar in der oberen linken Ecke (`new Point(0, 0)`, dritter Parameter).

Nun zum spannenden Teil: Die `threshold`-Methode erlaubt es, Pixelwerte abhängig von den Pixelwerten eines anderen `BitmapData`-Objekts zu verändern. Dabei geht die `threshold`-Methode alle Pixel des `Map-BitmapData`-Objekts durch, prüft, ob das Pixel eine bestimmte Bedingung erfüllt oder nicht, und verändert entsprechend die Pixel des `BitmapData`-Objekts, auf das die `threshold`-Funktion angewendet wird.

Die Bedingung können Sie dabei selbst bestimmen. In unserem Fall wird überprüft, ob ein Pixel der `Map-BitmapData` (`grey_bd`) schwarz ist (`"=="`, `0x000000`). Wenn dies so ist, wird das zugehörige Pixel in `output_bd` mit der Farbe Weiß (`0xFFFFFFFF`) gefüllt. Wenn dies nicht so ist, bleibt das Pixel unberührt – das zuvor kopierte Pixel in `original_water_bd` bleibt erhalten.

Exportieren Sie nun Ihren Flash-Film, und zeichnen Sie wieder ein wenig auf der linken Seite: Die entsprechenden Bereiche auf der rechten Seite werden sichtbar.



◀ **Abbildung 10**

Flächen, die durch das »Malen« mit der Maus in der Map (rechts) einen anderen Farbwert als Schwarz bekommen, werden im Ausgabe-Bitmap (links) freigelegt.

Damit der Eindruck von Flüssigkeit noch realistischer wird, soll die Flüssigkeit langsam wieder verdunsten, nachdem sie mit der Maus »aufgemalt« wurde. Um das zu erreichen, muss sich die sichtbare Fläche nach und nach verkleinern. Ergänzen Sie die `Main`-Funktion über den Event-Listenern aus Schritt 10 wie folgt:

```
reducecolor = new ColorTransform(.97, .97, .97, 1, 0,0,0,0);
```

Sie erstellen hier ein neues `ColorTransform`-Objekt, das die Farbe der Map um den Faktor 0.97 in Richtung Schwarz reduziert.

Fügen Sie nun folgenden Code in die `onStageEnterFrame`-Funktion noch vor die beiden Anweisungen aus dem vorigen Schritt ein:

```
grey_bd.colorTransform(grey_bd.rect, reducecolor);  
grey_bd.unlock();
```

Schritt 14

Verdunstungseffekt erstellen



Zwischenergebnis:
12_FLUESSIGKEIT •
MAIN_STEP14.AS

Nun wird in jedem Frame die Farbe der Map immer weiter abgeschwächt.

Testen Sie nun wiederum Ihren Film. Die sichtbare Fläche wird nun nach und nach immer kleiner. Später wird es so aussehen, als ob der Wassertropfen langsam verdunstet.

Schritt 15

Plastischer machen

Zwischenergebnis:

12_FLUESSIGKEIT •
MAIN_STEP15.AS



Aber irgendetwas fehlt immer noch. Um den Eindruck von Wasser komplett zu machen, fügen Sie noch einen BevelFilter hinzu. Fügen Sie folgenden Code in Ihre `Main`-Funktion über die Event-Listener aus Schritt 10 ein:

```
bevel = new BevelFilter();  
bevel.blurX = 10;  
bevel.blurY = 10;
```

Diese Zeilen erstellen einen neuen BevelFilter mit einer horizontalen und vertikalen Weichzeichnung von 10 Pixel.

Ergänzen Sie die `onStageEnterFrame`-Methode um folgende Zeile:

```
output_bd.applyFilter(output_bd, output_bd.rect, new Point(0, 0),  
bevel);
```

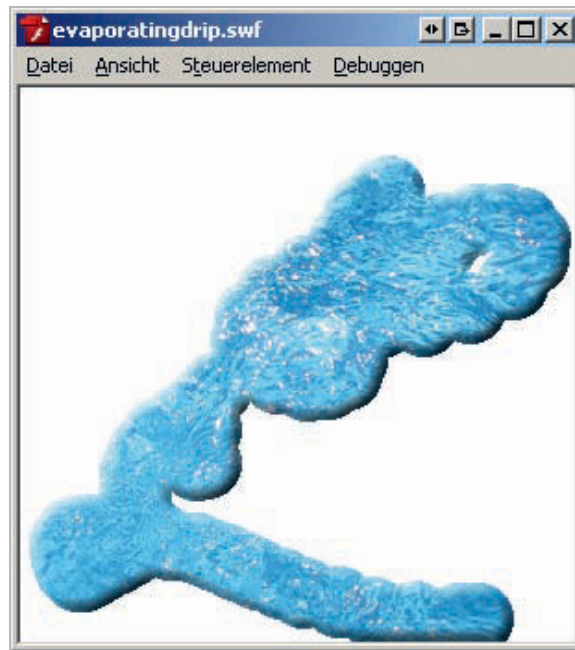


Abbildung 11 ►

Wie aus dem Nichts entsteht ein Wasserspritzer, der anschließend langsam verdunstet.

Feuer-Effekt

Ein feuriges Flash-Vergnügen mit den Bitmap-Operationen von Flash

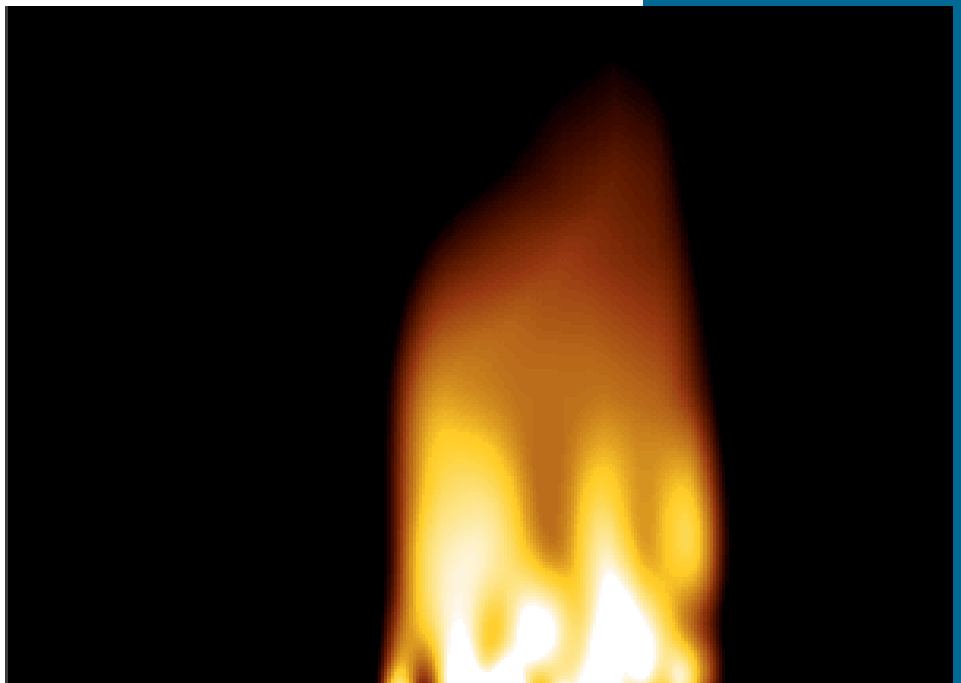
Oft gewünscht und selten erreicht: ein Feuereffekt in Flash. In früheren Versionen wurden viele Versuche unternommen, diesen Effekt zu simulieren. Meist litt jedoch die Performance sehr darunter. Dank des BitmapData-Objekts und ActionScript 3 haben Sie dieses Problem nun nicht mehr: Lernen Sie in diesem Workshop, wie mit den Bitmap-Funktionen von Flash ein eindrucksvoller Feuer-Effekt entsteht.

Zielsetzungen:

- BitmapData-Objekt anwenden
- Maske erstellen
- BlurFilter einsetzen
- paletteMap-Funktion nutzen

[Ergebnis: 13_FEUER-EFFEKT
 • FEUEREFFEKT.SWF]

```
package
{
    import flash.display.MovieClip;
    import flash.display.BitmapData;
    import flash.display.Bitmap;
    import flash.filters.BlurFilter;
    import flash.geom.Rectangle;
    import flash.geom.Point;
    import flash.events.Event;
    public class Main extends MovieClip
    {
        private var grey_bd:BitmapData;
        private var flame_bd:BitmapData;
        private var gradient_bd:BitmapData;
        private var output_bmp:Bitmap;
        private var map_bmp:Bitmap;
```



Schritt 1 Dokument erstellen

Erstellen Sie ein neues AS3-Flash-Dokument mit der Größe 600x300 Pixel und einer Bildrate von 25 bps. Speichern Sie dieses Dokument in einem neuen Ordner unter dem Namen FEUEREFFEKT.FLA ab.



Abbildung 1 ►
Dokumenteigenschaften für
das ActionScript 3-Flash-
Dokument

Schritt 2 Farbverlauf für das Feuer

Sie benötigen später in diesem Workshop einen Verlauf, aus dem die Farbwerte für das Feuer extrahiert werden. Legen Sie dafür mit Hilfe des Rechteckwerkzeuges auf der Bühne ein Quadrat mit einer Höhe von 256 Pixel und einer Breite von 129 Pixel an. Die Breite spielt eigentlich keine Rolle – doch dazu später mehr.

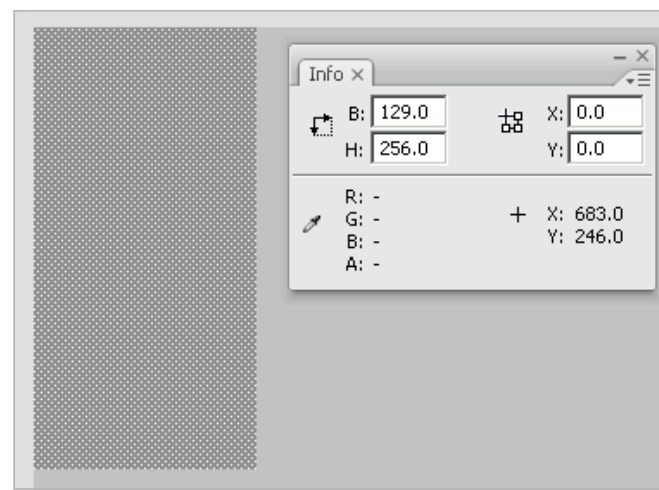
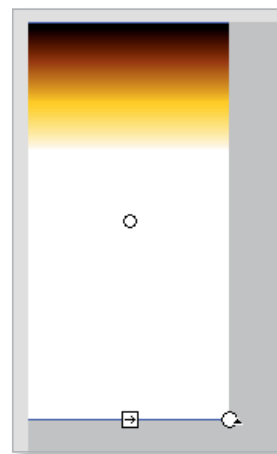
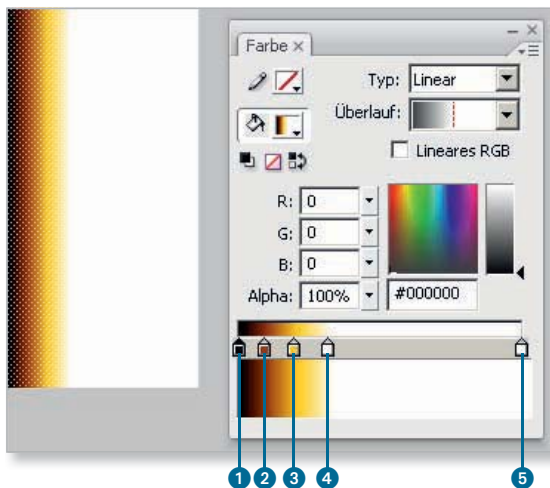


Abbildung 2 ►
Ein graues Quadrat bildet
den Ausgangspunkt für den
Farbverlauf des Feuers.

Markieren Sie nun das eben erstellte Quadrat, und öffnen Sie über FENSTER • FARBE die Palette FARBE. Wählen Sie im Pulldown-Menü TYP den Eintrag »Linear«.

Klicken Sie nun nacheinander auf die Farbeimer, und geben Sie dem Verlauf folgende Werte:

- ▶ #000000 ❶
- ▶ #993300 ❷
- ▶ #FFCC00 ❸
- ▶ #FFFFFF ❹ und ❺



◀ ◀ **Abbildung 3**

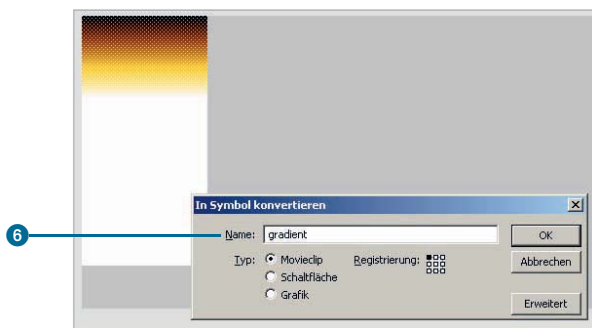
Farben des Farbverlaufs festlegen

◀ **Abbildung 4**

Der Verlauf wird mit Hilfe des Farbverlaufswerkzeugs gedreht.

Drehen Sie den Verlauf nun mit Hilfe des Farbverlaufswerkzeug  so, dass das Schwarz oben ist.

Markieren Sie den Farbverlauf nun erneut, und konvertieren Sie ihn mit MODIFIZIEREN • IN SYMBOL KONVERTIEREN oder **F8** in ein neues Symbol. Geben Sie ihm in dem sich nun öffnenden Dialog den Namen »gradient« ❻.



Schritt 3

Farbverlauf in Movieclip konvertieren

◀ **Abbildung 5**

Konvertieren Sie den Verlauf in einen Movieclip.

Nachdem Sie das Symbol erstellt haben, müssen Sie noch einen Instanznamen vergeben, damit der Movieclip auch von Ihrem Skript angesprochen werden kann. Geben im Eigenschaftsfenster »gradient_mc« **1** ein.

Abbildung 7 ►

Geben Sie dem Movieclip den Instanznamen »gradient_mc«.



Schritt 4 Dokumentklasse erstellen

Zwischenergebnis:

13_FEUER-EFFEKT •
MAIN_STEP04.AS



Erstellen Sie nun eine Dokumentklasse. Wählen Sie dafür den Menüpunkt DATEI • NEU • ACTIONSCRIPT-KLASSE. Fügen Sie die folgenden Zeilen der Klasse hinzu – es ist das Grundgerüst, das Sie später erweitern werden:

```
1: package
2: {
3:     import flash.display.MovieClip;
4:     import flash.display.BitmapData;
5:     import flash.display.Bitmap;
6:     import flash.filters.BlurFilter;
7:     import flash.geom.Rectangle;
8:     import flash.geom.Point;
9:     import flash.events.Event;
10:    public class Main extends MovieClip
11:    {
12:        private var grey_bd:BitmapData;
13:        private var flame_bd:BitmapData;
14:        private var gradient_bd:BitmapData;
15:        private var output_bmp:Bitmap;
16:        private var map_bmp:Bitmap;
17:        private var blur:BlurFilter;
18:        private var map:Array;
19:        public function Main()
20:        {
21:        }
22:    }
23: }
```

In den Zeilen 3 bis 9 importieren Sie die Klassen, die für die Erstellung des Feuer-Effekts benötigt werden. Dann werden ab Zeile 12 die Variablen deklariert, die Sie für den Effekt benötigen. Speichern Sie diese Datei unter dem Namen MAIN.AS in demselben Verzeichnis wie die fla-Datei.

Damit der Flash-Film auch »weiß«, dass er eine Dokumentklasse hat, geben Sie im Feld DOKUMENTKLASSE »Main« ein.



Sie benötigen den zuvor erstellten Verlaufs-Movieclip für Ihr Skript, er soll aber auf der Bühne so nicht sichtbar sein. Fügen Sie deshalb folgende Zeile in die Main-Funktion der Klasse ein:

```
gradient_mc.visible = false;
```

Für die Bewegung des Feuers benötigen Sie ein BitmapData-Objekt, in dem die Feuerbewegung simuliert wird.

Fügen Sie Ihrem Skript unter die Codezeile aus Schritt 6 folgenden Code hinzu:

```
1: grey_bd = new BitmapData(300,300,false,0);
2: map_bmp = new Bitmap();
3: map_bmp.bitmapData = grey_bd;
4: addChild(map_bmp);
5: map_bmp.x = 300;
```

In Zeile 1 erstellen Sie das BitmapData-Objekt `grey_bd`. Es soll eine Breite und Höhe von jeweils 300 Pixel haben und keine Transparenzen unterstützen (`false`).

Eigentlich wird `grey_bd` nur intern für das Skript benötigt. Zur Verdeutlichung werden Sie es aber auch auf der Bühne darstellen. Dafür erzeugen Sie in Zeile 2 ein neues Bitmap Objekt, `map_bmp`. Dieses weisen Sie in Zeile 3 an, die Bitmapdaten von `grey_bd` darzustellen. In Zeile 4 machen Sie es auf der Bühne sichtbar und positionieren es anschließend (in Zeile 5) auf den x-Wert 300 Pixel.

Wenn Sie den Flash-Film nun exportieren, werden Sie auf der rechten Seite ein 300 x 300 Pixel großes Quadrat sehen.

Schritt 5

Dokumentklasse verknüpfen

◀ Abbildung 8
Dokumentklasse verknüpfen

Schritt 6

Verlauf unsichtbar machen

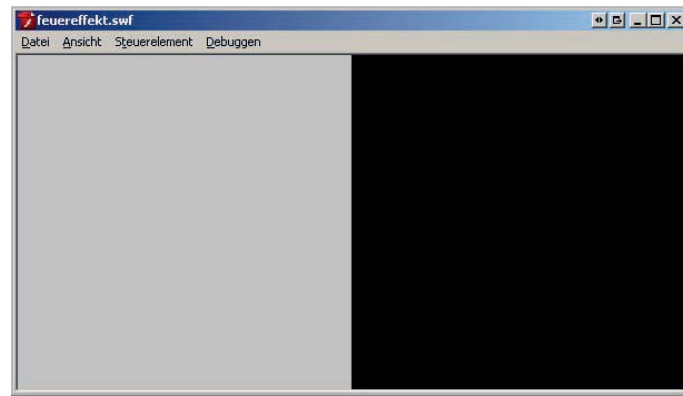
Schritt 7

BitmapData-Objekt erstellen



Zwischenergebnis:
13_FEUER-EFFEKT •
MAIN_STEP07.AS

Abbildung 9 ►
Die gerade erstellte Ausgabe-Bitmap, schwarz auf der rechten Seite



Schritt 8 Bewegungs-Blur erstellen

Zwischenergebnis:
13_FEUER-EFFEKT •
MAIN_STEP08.AS 

Die Bewegung der Map (die später als Berechnungsgrundlage für das Ergebnis dienen wird) wird unter anderem durch einen `BlurFilter` erzeugt. Ergänzen Sie deshalb unter die Codezeilen aus Schritt 7 folgende Anweisung:

```
blur = new BlurFilter(4,10);
```

Da Flammen ja nach oben steigen, hat der `BlurFilter` eine horizontale Weichzeichnung von lediglich 4, aber eine vertikale Weichzeichnung von 10.

Schritt 9 Map animieren

Zwischenergebnis:
13_FEUER-EFFEKT •
MAIN_STEP09.AS 

Nun werden Sie die Map animieren. Dafür müssen Sie sich zunächst in der nächsten Zeile von `Main()` beim `ENTER_FRAME`-Event anmelden:

```
addEventListener(Event.ENTER_FRAME, onStageEnterFrame);
```

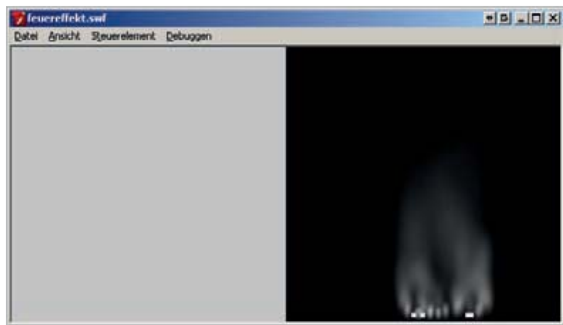
In jedem Frame wird ab jetzt die Funktion `onStageEnterFrame()` aufgerufen. Diese erstellen Sie nun. Fügen Sie dazu folgende Codezeilen unter der `Main()` hinzu:

```
1: private function onStageEnterFrame(evt:Event):void
2: {
3:     grey_bd.applyFilter(grey_bd, grey_bd.rect, new
        Point(0,-2), blur);
4:     for ( var i:int = 0; i < 4 ; i ++ ) {
5:         grey_bd.fillRect(new Rectangle(Math.random()*100+120,
            290, 5, 5),0xFFFFFFFF);
6:     }
7: }
```

In Zeile 3 wird zunächst der `BlurFilter` auf das `BitmapData`-Objekt `grey_bd` angewendet. `new Point(0, -2)` bewirkt, dass die Bildinformationen in jedem Frame um zwei Pixel nach oben verschoben werden. So entsteht der Eindruck, die Flammen bewegten sich nach oben.

Nun zeichnen Sie in Zeile 5 Quadrate (`fillRect()`), welche die Map mit Pixeln füllen. Diese Quadrate sollen weiß sein (`0xFFFFFFFF`). Die Breite und Höhe von 5 Pixel (`5, 5`) übergeben Sie in Parameter 3 und 4 des `Rectangle`-Objekts. In Parameter 2 des `Rectangle`-Objekts geben Sie an, dass das Quadrat auf einer Höhe von 290 erzeugt werden soll. Parameter 1 ist nicht minder wichtig. Hier geben Sie an, dass das Quadrat auf einer horizontalen Zufallsposition zwischen den Werten 120 und 220 erzeugt werden soll (`Math.random()*100+120`).

Wenn Sie Ihr Dokument nun exportieren, sehen Sie, dass weiße Quadrate erzeugt, weichgezeichnet und nach oben verschoben werden.



◀ **Abbildung 10**

Die Flammen lodern schon – wenn auch in Schwarzweiß.

Bisher sieht die Map ja ein bisschen wie ein kleines Feuerchen aus. Um daraus ein richtiges Feuer zu machen, führen Sie das Skript von eben einfach dreimal pro `ENTER_FRAME` aus. Ersetzen Sie Ihre `onStageEnterFrame`-Funktion durch folgende:

```
1: private function onStageEnterFrame(evt:Event):void
2: {
3:     for ( var r:int = 0; r < 3 ; r++ ) {
4:         grey_bd.applyFilter(grey_bd, grey_bd.rect, new
           Point(0,-2), blur);
5:         for ( var i:int = 0; i < 4 ; i ++ ) {
6:             grey_bd.fillRect(new Rectangle(Math.random()*100+120,
           290, 5, 5),0xFFFFFFFF);
7:         }
8:     }
9: }
```

Schritt 10

Animation verstärken



Zwischenergebnis:
13_FEUER-EFFEKT •
MAIN_STEP10.AS

In Zeile 3 wurde eine `for`-Schleife eingefügt, wodurch der Code dreimal ausgeführt wird. Wenn Sie Ihr Dokument jetzt noch einmal exportieren, hat die Animation schon deutlich mehr »Drive«.

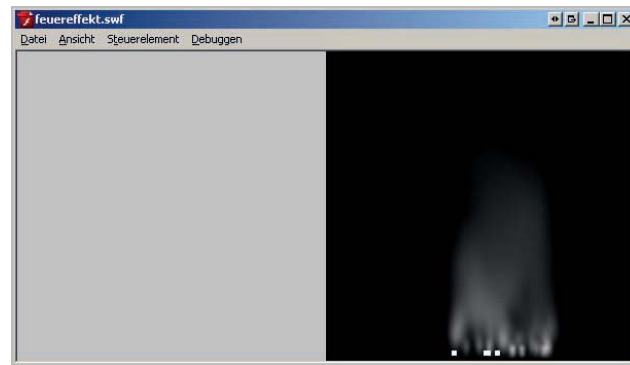


Abbildung 11 ►

Aus dem kleinen Flämmchen ist ein Feuer geworden.

Schritt 11

BitmapData-Objekt für die Ausgabe

Zwischenergebnis:

13_FEUER-EFFEKT •
MAIN_STEP11.AS



Nun machen Sie sich an die Ausgabe der Flamme. Ergänzen Sie die `Main`-Funktion über dem Event-Listener aus Schritt 9 wie folgt:

```
1: flame_bd = new BitmapData(300,256,false,0xFF0000);
2: output_bmp = new Bitmap();
3: output_bmp.bitmapData = flame_bd;
4: addChild(output_bmp);
```

Das Ausgabe-`BitmapData`-Objekt `flame_bd` erstellen Sie in Zeile 1. Es soll eine Breite von 300 Pixel und eine Höhe von 256 Pixel haben. Es unterstützt keine Transparenzen (`false`) und hat den Anfangsfarbwert Rot (`0xFF0000`).

Um die Bitmapdaten anzuzeigen, erstellen Sie in Zeile 2 eine `Bitmap`, die Sie in Zeile 3 anweisen, die Bitmapdaten von `flame_bd` anzuzeigen. Anschließend (in Zeile 4) machen Sie es mit Hilfe von `addChild()` auf der Bühne sichtbar.

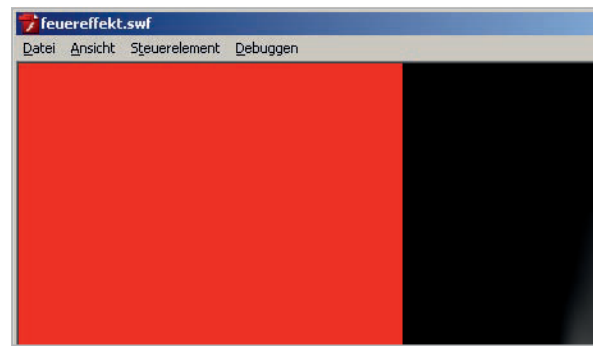


Abbildung 12 ►

Das `Bitmap`-Objekt, das die finale Animation anzeigen wird, rot auf der linken Seite

Nun werden Sie die Farbwerte, die später in der Flamme angezeigt werden, aufnehmen. Damit Ihr zuvor auf der Bühne erstellter Verlauf für Ihr Skript verfügbar wird, müssen Sie ein `BitmapData`-Objekt erstellen. Ergänzen Sie Ihren Code direkt unter den eben in Schritt 11 eingefügten Codezeilen:

```
gradient_bd = new BitmapData(1, 256, false, 0);
gradient_bd.draw(gradient_mc);
```

Ihr zuvor auf der Bühne erstellter Verlauf hat eine Breite von 300 Pixel. Sie benötigen allerdings nur die vertikalen Farbinformationen. Deshalb reicht es auch, dass das `BitmapData`-Objekt (`gradient_bd`) eine Breite von nur einem Pixel hat. Die Höhe soll 256 betragen, da Sie alle 256 Pixel des Farbverlaufs benötigen.

In der zweiten Zeile nehmen Sie die Bitmapdaten des Movieclips `gradient_mc` auf.

Später werden Sie die Farben der Map (`grey_bd`) mit Farben des Verlaufes ersetzen. Dafür müssen die Farbinformationen des Verlaufes in einem Array gespeichert werden. Fügen Sie Ihrem Skript – wieder direkt unter den eben eingefügten Codezeilen – folgende Zeilen hinzu:

```
1: map = new Array(256);
2: for(var i:int = 0; i < 256; i++) {
3:     map[i] = gradient_bd.getPixel(0, i);
4: }
```

In Zeile 1 erstellen Sie ein neues Array `map`, das 256 Einträge umfassen soll.

Nun werden Sie die Farbwerte des Verlaufes von oben nach unten durchgehen. Dafür wird in Zeile 2 eine `for`-Schleife 256 Mal aufgerufen.

In Zeile 3 lesen Sie den Farbwert des jeweiligen Pixels `i` mit `getPixel()` aus und speichern diesen in Ihr zuvor erstelltes Array. Nun befindet sich ein Abbild des Verlaufes in Zahlenform in Ihrem Array `map`.

Jetzt zum spannenden Teil. Sie werden nun die Farbwerte von `grey_bd` durch die eben im Array `map` gespeicherten Farbwerte ersetzen. Fügen Sie Ihrem Code unter der `for`-Schleife in der `onStageEnterFrame`-Funktion folgende Zeile hinzu:

Schritt 12

Verlauf in BitmapData-Objekt speichern



Zwischenergebnis:
13_FEUER-EFFEKT •
MAIN_STEP12.AS

Schritt 13

Verlaufsfarbe in Array speichern



Zwischenergebnis:
13_FEUER-EFFEKT •
MAIN_STEP13.AS

Schritt 14

Flammen erzeugen

Zwischenergebnis:
13_FEUER-EFFEKT •
MAIN_STEP14.AS



```
flame_bd.paletteMap(grey_bd, new Rectangle(0, 20, 300, 256), new  
    Point(0, 0), map, [0], [0], null);
```

Die Methode `paletteMap()` ersetzt Farbwerte eines `BitmapData`-Objekts abhängig von einem Quellbild mit den Farbwerten aus einem Array.

Das Quellbild (Parameter 1) ist `grey_bd`. Im zweiten Parameter geben Sie an, welche Fläche von `grey_bd` für die Berechnung herangezogen werden soll. Im ganz unteren Teil von `grey_bd` befinden sich ja die noch nicht weichgezeichneten, erzeugten Quadrate. Darum übergeben wir dem `Rectangle`-Objekt lediglich eine Höhe von 256, und nicht 300, was ja die eigentliche Höhe von `grey_bd` ist. Alles darunter wird ignoriert.

In den Parametern 4, 5, 6 und 7 der `paletteMap`-Methode spielt die Musik. Jedem Farbkanal kann hier ein eigenes Array zugewiesen werden. Wir beschränken uns jedoch auf den roten Farbkanal (Parameter 4, `map`).

`paletteMap()` extrahiert jetzt aus jedem Pixel von `grey_bd` den roten Farbwert und ersetzt ihn durch den entsprechenden Eintrag im Array. Wird also ein Farbwert von 5 festgestellt, wird dieser Wert durch die Zahl des fünften Eintrags des Arrays ersetzt. Stellt die Funktion einen Wert von 244 fest, wird entsprechend der 244. Eintrag des Arrays herangezogen.

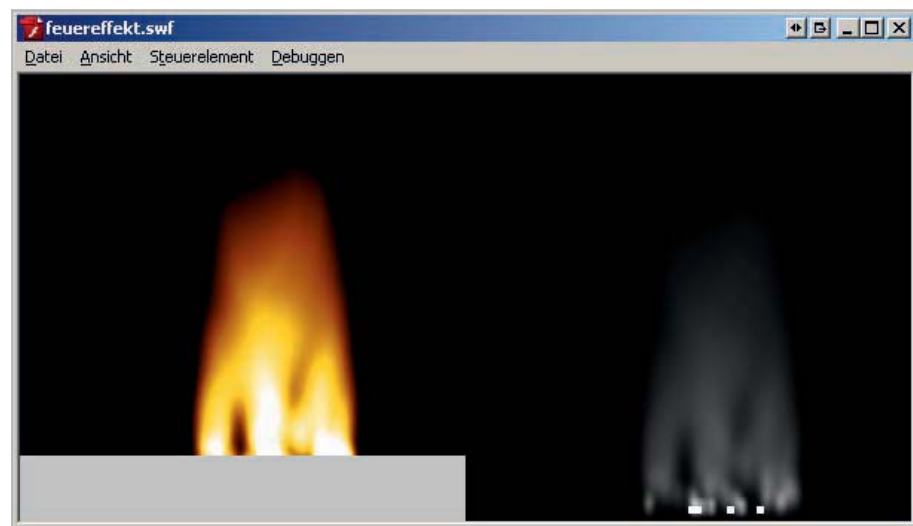


Abbildung 13 ►

Exportieren Sie nun Ihr Dokument, und lassen Sie es zündeln!

Web 2.0-Reflexion

Movieclips mit einem nahtlosen Übergang und Alphaverlauf spiegeln

Apple hat damit angefangen, und alle haben sie kopiert: die Spiegelung mit Alphaverlauf. In diesem Workshop lernen Sie, eine Spiegelung dynamisch auf einen Movieclip anzuwenden, auch wenn sich in dem Movieclip animiertes Material befindet.

Zielsetzungen:

BitmapData-Objekt anwenden

Mit einer Matrix arbeiten

Einen Alphaverlauf dynamisch erstellen

[Ergebnis: 14_WEB2.0-REFLEXION • SPIEGELEFFECT.SWF]

```
package
{
    import flash.display.Sprite;
    import flash.display.MovieClip;
    import flash.display.BitmapData;
    import flash.display.Bitmap;
    import flash.display.Shape;
    import flash.geom.Point;
    import flash.events.Event;
    import flash.geom.Matrix;
```



Schritt 1 Dokument erstellen

Erstellen Sie ein neues AS3-Flash-Dokument mit den Grösse 280x418 Pixel und einer BILDRATE von 25 bps. Speichern Sie dieses Dokument unter dem Namen REFLECTION.FLA ab.



Abbildung 1 ►
Über MODIFIZIEREN • DOKUMENT legen Sie die Dokumenteigenschaften fest.

Schritt 2 Quell-Movieclip erstellen

Erstellen Sie zunächst einen Movieclip, der gespiegelt werden soll. Importieren Sie dazu ein beliebiges Bild, oder erstellen Sie mit Hilfe der Flash-Werkzeuge eine Grafik. Beachten Sie, dass Ihr Movieclip höchstens 280 x 209 Pixel groß sein sollte, weil die Reflexion bei der eingestellten Bühnengröße sonst nicht mehr komplett sichtbar ist. In diesem Beispiel nutzen wir das Foto IMG.JPG.

Konvertieren Sie Ihre Grafik(en) mit MODIFIZIEREN • IN SYMBOL KONVERTIEREN oder **[F8]** in ein Symbol. Geben Sie dem Movieclip in dem sich nun öffnenden Dialog den Namen »img« **1**. Achten Sie darauf, dass die REGISTRIERUNG links oben **2** liegt. Anschließend vergeben Sie im Eigenschaftenfenster den Instanznamen »img« **3**.

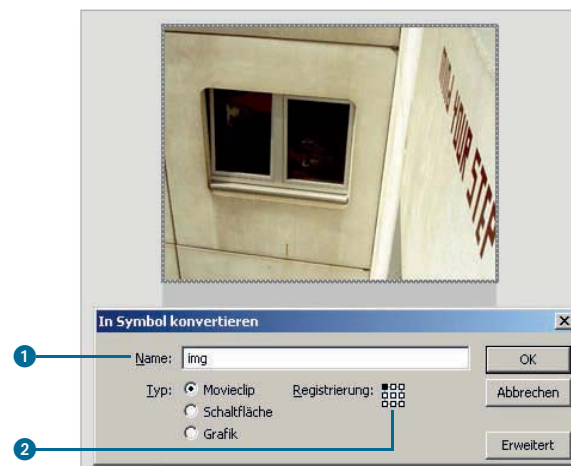
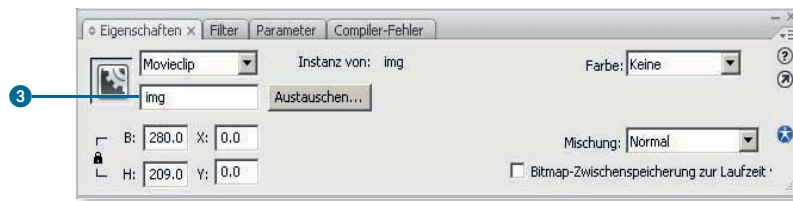


Abbildung 2 ►
Das Foto wird in einen Movieclip konvertiert.

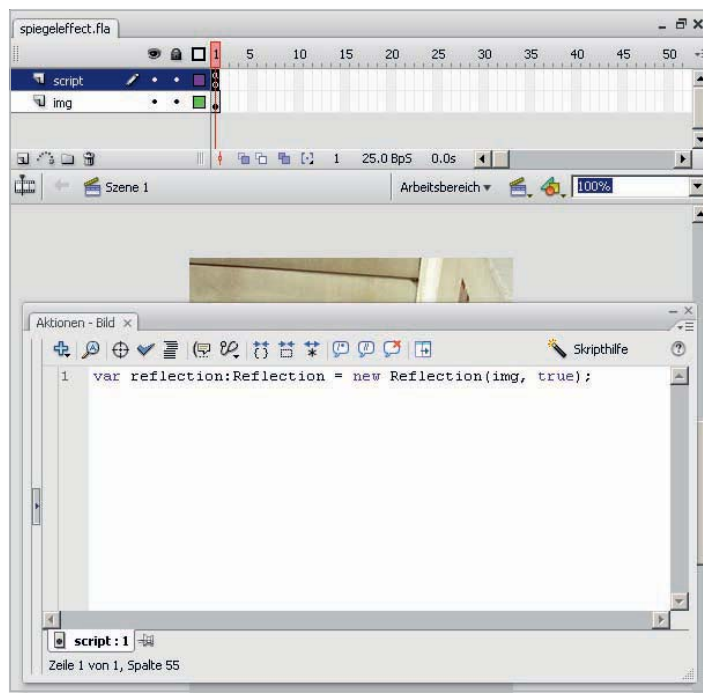


◀ **Abbildung 3**
Der Foto-Movieclip bekommt den Instanznamen »img«.

Im Flash-Film selbst wird zunächst die (noch zu erstellende) Reflexions-Klasse dynamisch von der Zeitleiste aus aufgerufen. Die Reflexions-Klasse wird also nicht als Dokumentklasse erstellt oder fest mit einem Movieclip verbunden, sondern kann von jeder Zeitleiste eines (noch so verschachtelten) Movieclips instantiiert werden.

Erstellen Sie auf der Hauptzeitleiste eine neue Ebene mit dem Namen »script«. Öffnen Sie über FENSTER • AKTIONEN oder **F9** bzw. **Alt** + **F9** den ActionScript-Editor. Fügen Sie hier folgende Zeile hinzu:

```
var reflection:Reflection = new Reflection(img, true);
```



◀ **Abbildung 4**
Die Reflexions-Klasse wird aufgerufen.

Erstellen Sie über DATEI • NEU • ACTIONSCRIPT-DATEI eine neue Klasse. Speichern Sie diese über DATEI • SPEICHERN UNTER... als REFLECTION.AS in demselben Ordner wie die fla-Datei ab.

Schritt 4 Reflexions-Klasse anlegen

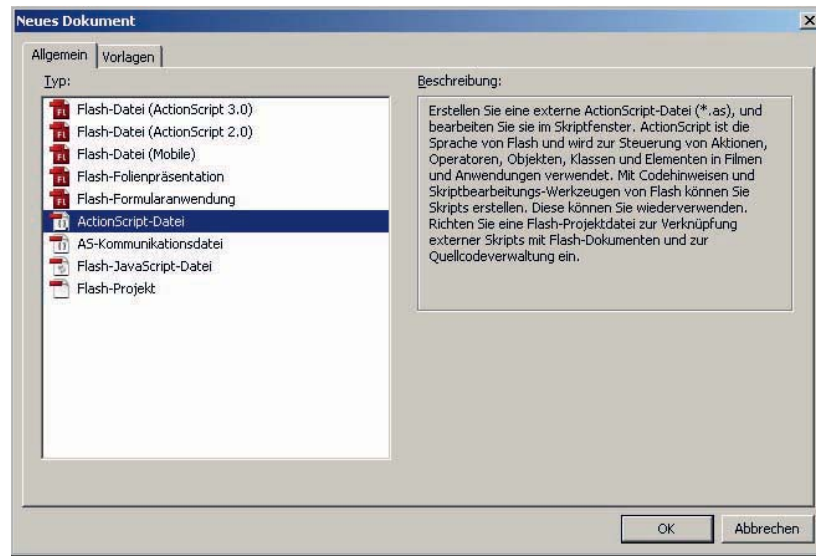


Abbildung 5 ►
Eine neue ActionScript-Datei
anlegen

Schritt 5 Klassengrundgerüst

Zwischenergebnis:
14_WEB2.0-REFLE-
XION • REFLECTION_
STEP05.AS



Fügen Sie folgenden Code in die neu erstellte Klasse ein:

```
1: package
2: {
3:     import flash.display.Sprite;
4:     import flash.display.MovieClip;
5:     import flash.display.BitmapData;
6:     import flash.display.Bitmap;
7:     import flash.display.Shape;
8:     import flash.geom.Point;
9:     import flash.events.Event;
10:    import flash.geom.Matrix;
11:    public class Reflection extends Sprite
12:    {
13:        private var w:Number;
14:        private var h:Number;
15:        private var img:MovieClip;
16:        private var original_bd:BitmapData;
17:        private var rotate_matrix:Matrix;
18:        private var rotate_bd:BitmapData;
19:        private var output_bd:BitmapData;
20:        private var gradient_bd:BitmapData;
21:        private var display_bmp:Bitmap;
22:        public function Reflection(mc:MovieClip, video:Boolean
           = false)
23:        {
```

```

24:         }
25:     }
26: }

```

In den Zeilen 3 bis 10 importieren Sie die für das Skript benötigten Klassen. Die erforderlichen Variablen deklarieren Sie in den Zeilen 13 bis 21.

Dem Konstruktor in Zeile 22 wird bei Aufruf mindestens ein Parameter übergeben, nämlich der Movieclip, der reflektiert werden soll. Als zweiter optionaler Parameter `video` kann festgelegt werden, ob sich in dem Movieclip ein Video oder animierter Inhalt befindet, der in jedem Frame ausgelesen werden muss.

Mit `video:Boolean = false` geben Sie im Konstruktor an, dass der Parameter, wenn er nicht übergeben wird, `false` ist.

Fügen Sie im Konstruktor nun folgenden Code hinzu:

```

img = mc;
w = mc.width;
h = mc.height;

```

Für die Erstellung des Spiegeleffektes benötigen Sie später die Pixelinformationen des Ausgangs-Movieclips. In der ersten Zeile speichern Sie eine Referenz auf den im Konstruktor als ersten Parameter übergebenen Movieclip ab.

Für die Berechnungen benötigen Sie die Größe des zu reflektierenden Movieclips. Dessen Breite und Höhe speichern Sie in den beiden folgenden Zeilen.

Um die Bitmapdaten des Quellbildes für das Skript verfügbar zu machen, schreiben Sie sie in ein BitmapData-Objekt. Ergänzen Sie Ihren Code dazu direkt unter den in Schritt 6 eingefügten Codezeilen wie folgt:

```

original_bd = new BitmapData(w, h);
original_bd.draw(mc);

```

In der ersten Zeile erzeugen Sie das BitmapData-Objekt `original_bd`, das die Größe des Quell-Movieclips hat.

Ein Abbild des Quell-Movieclips zeichnen Sie in der zweiten Zeile in die Variable `original_bd`.

Schritt 6

Referenz und Größe speichern



Zwischenergebnis:
14_WEB2.0-REFLE-
XION • REFLECTION_
STEP06.AS

Schritt 7

Quellbild als BitmapData speichern



Zwischenergebnis:
14_WEB2.0-REFLE-
XION • REFLECTION_
STEP07.AS

Schritt 8

Rotationsmatrix erstellen

Zwischenergebnis:

14_WEB2.0-REFLE-
XION • REFLECTION_
STEP08.AS



Natürlich muss die Spiegelung des Bildes auf dem Kopf stehen. Zu diesem Zweck erstellen Sie ein `Matrix`-Objekt. Ergänzen Sie im Konstruktor folgende Codezeilen:

```
rotate_matrix = new Matrix();  
rotate_matrix.scale(1, -1);  
rotate_matrix.translate(0, h);
```

Zuerst erstellen Sie das neue `Matrix`-Objekt `rotate_matrix`.

In der nächsten Zeile skalieren Sie dieses mit dem Faktor `-1` in der Vertikalen. Die Matrix wird also »auf den Kopf« gestellt.

Anschließend müssen Sie die Matrix mit Hilfe von `translate` um die Höhe des Quell-Movieclips in der Vertikalen nach unten verschieben. Ansonsten läge sie genau über dem Quell-Movieclip – und eine Spiegelung liegt ja immer unter einem Objekt.

Schritt 9

Rotations-BitmapData erstellen

Zwischenergebnis:

14_WEB2.0-REFLE-
XION • REFLECTION_
STEP09.AS



Nun drehen Sie die `original_bg`. Ergänzen Sie folgende Zeilen ganz unten im Konstruktor:

```
rotate_bd = new BitmapData(w, h, false, 0);  
rotate_bd.draw(original_bd, rotate_matrix);
```

In der ersten Zeile erstellen Sie ein neues `BitmapData`-Objekt mit den Maßen des Ausgangsbildes.

Jetzt zeichnen Sie `original_bd` unter Angabe der `rotate_matrix` in `rotate_bd`. In `rotate_bd` liegt das Bild jetzt in gedrehter und verschobener Version vor.

Schritt 10

BitmapData-Objekt für die Ausgabe erstellen

Zwischenergebnis:

14_WEB2.0-REFLE-
XION • REFLECTION_
STEP10.AS



Fügen Sie Ihrem Skript folgende Zeilen hinzu (wieder ganz unten im Konstruktor, unter den eben eingefügten Codezeilen):

```
output_bd = new BitmapData(w, h, true, 0);  
output_bd.copyPixels(rotate_bd, rotate_bd.rect, new Point(0, 0));
```

Zuerst erstellen Sie ein neues `BitmapData`-Objekt `output_bd`, das die Breite und Höhe des Quell-Movieclips hat. Der dritte Parameter (`true`) legt fest, dass das `BitmapData`-Objekt Transparenzen unterstützen soll, da Sie der Spiegelung später einen transparenten Verlauf hinzufügen werden.

Anschließend kopieren Sie die Bilder in der zweiten Zeile von `original_bd` in `output_bd`.

Nun benötigen Sie noch ein Bitmap-Objekt, das die Darstellung der Bild-daten übernimmt. Ergänzen Sie Ihren Code wie folgt:

```
1: display_bmp = new Bitmap();
2: display_bmp.bitmapData = output_bd;
3: display_bmp.y = h;
4: addChild(display_bmp);
```

Erstellen Sie dieses Bitmap-Objekt in der ersten Zeile, und weisen Sie es in Zeile 2 an, die Bitmapdaten von `output_bd` anzuzeigen.

Außerdem verschieben Sie `display_bmp` in Zeile 3 auf die y-Position, die exakt der Unterkante des Quell-Movieclips entspricht.

Mit `addChild()` in Zeile 4 sorgen Sie dafür, dass das Bitmap-Objekt auf der Bühne dargestellt wird.

Nun muss noch die Reflexions-Klasse selbst angezeigt werden. Ergänzen Sie (direkt unter den eben eingefügten Codezeilen) folgende Anweisung:

```
mc.parent.addChild(this);
```

Mit dem `addChild`-Befehl, der in derselben Hierarchie-Ebene wie der Quell-Movieclip aufgerufen wird (`mc.parent`), zeigen Sie `display_bmp` auf der Hauptzeitleiste an.

Wenn Sie Ihr Dokument nun testen, wird unterhalb des Quell-Movieclips die Reflexion angezeigt (siehe Abbildung 6).

Die Reflexion soll einen Verlauf beinhalten, der von 100 % Alpha (also voll sichtbar) nach 0 % (unsichtbar) ausfadet.

Dazu erstellen Sie zunächst eine Funktion, die Ihnen einen Verlauf zurückgibt. Ergänzen Sie Ihr Skript unter dem Konstruktor um folgenden Code:

Schritt 11

Ausgabe-Bitmap erstellen



Zwischenergebnis:
14_WEB2.0-REFLE-
XION • REFLECTION_
STEP11.AS

Schritt 12

Reflexions-Klasse anzeigen



Zwischenergebnis:
14_WEB2.0-REFLE-
XION • REFLECTION_
STEP12.AS

Schritt 13

Flash-Film testen

Schritt 14

Alphaverlauf hinzufügen



▲ Abbildung 6

Das Bild wird gespiegelt.

Zwischenergebnis:

14_WEB2.0-REFLE-
XION • REFLECTION_
STEP14.AS



```

1: private function createGradientBitmapData(w:Number,
    h:Number):Shape
2: {
3:     var colors:Array = [0x000000, 0x000000];
4:     var fillType:String = "linear";
5:     var alphas:Array = [100, 0];
6:     var ratios:Array = [0, 0xFF];
7:     var rot:Number = 90 * Math.PI / 180;
8:     var matrix:Matrix = new Matrix();
9:     matrix.createGradientBox(w, h, rot, 0, 0);
10:    var b:Shape = new Shape();
11:    b.graphics.beginGradientFill(fillType, colors, alphas, ra-
        tios, matrix);
12:    b.graphics.moveTo(0, 0);
13:    b.graphics.lineTo(w, 0);
14:    b.graphics.lineTo(w, h);
15:    b.graphics.lineTo(0, h);
16:    b.graphics.lineTo(0, 0);
17:    b.graphics.endFill();
18:    return b;
19: }

```

In Zeile 1 legen Sie den Funktionsnamen `createGradientBitmapData` fest. Die Methode erwartet die zwei Parameter `w` und `h`. Dabei handelt es sich um die Breite und die Höhe des zu erstellenden Verlaufes. Die Methode gibt ein `Shape`-Objekt zurück, das später von einem `BitmapData`-Objekt gezeichnet wird.

Die Farben des Verlaufes legen Sie in Zeile 3 fest. Sie sind beliebig, da Sie nur die Alphawerte des Verlaufes benötigen.

Den Typ definieren Sie in Zeile 4 als linearen Verlauf. Die Alphawerte legen Sie in Zeile 5 fest. Diese sollen von 100% bis 0% reichen.

Die Variable `ratios` in Zeile 6 gibt an, dass der gesamte Farbraum von 0 bis 255 genutzt werden soll.

Außerdem muss der Verlauf rotiert werden, da er sich standardmäßig von links nach rechts vollzieht. Die Rotationsvariable legen Sie in Zeile 7 fest. Sie muss im Bogenmaß angegeben werden, weshalb die Gradzahl 90 mit $\text{Pi}/180$ multipliziert werden muss.

In Zeile 8 erstellen Sie dann eine neue `Matrix`, der Sie in Zeile 9 die Flächenangaben für den Verlauf übergeben.

Den `Shape`, der von der Funktion zurückgegeben wird, erstellen Sie in Zeile 10.

Umrechnung

Sie können einen Winkel, der im Bogenmaß angegeben ist, wie folgt in das Gradmaß umrechnen:

$$\text{Grad} = \text{Radiant} * (180/\text{Pi})$$

Um einen Winkel, der im Gradmaß angegeben ist, in das Bogenmaß umzurechnen, verwenden Sie folgende Formel:

$$\text{Radiant} = \text{Grad} * (\text{Pi}/180)$$

In Zeile 11 weisen Sie den Shape an, eine Verlaufszeichnung zu beginnen, die die zuvor erstellten Parameter hat.

Die weiteren Zeilen 12 bis 17 sind für die Zeichnung des Verlaufs zuständig.

In Zeile 18 schließlich wird der soeben gezeichnete Verlauf als Shape-Objekt zurückgegeben.

Nun muss der Code zur Erstellung der Spiegelung folgendermaßen angepasst werden. Ersetzen Sie diesen Code aus Schritt 10:

```
output_bd = new BitmapData(w, h, true, 0);
output_bd.copyPixels(rotate_bd, rotate_bd.rect, new Point(0, 0));
```

mit folgenden Codezeilen:

```
1: gradient_bd = new BitmapData(w, h, true, 0);
2: gradient_bd.draw(createGradientBitmapData(w, h));
3: output_bd = new BitmapData(w, h, true, 0);
4: output_bd.copyPixels(rotate_bd, rotate_bd.rect, new Point(0,
    0), gradient_bd, new Point(0, 0), true);
5: display_bmp = new Bitmap();
6: ...
```

In Zeile 1 erstellen Sie ein BitmapData-Objekt mit Namen `gradient_bd`, das wieder die Breite und Höhe des Quell-Movieclips hat.

Innerhalb der `draw`-Methode in Zeile 2 rufen Sie `createGradientBitmapData()` auf und übergeben dieser Methode die Breite und Höhe. Das zurückgegebene Shape-Objekt wird in `gradient_bd` gespeichert.

Anschließend muss noch die `copyPixels`-Funktion in Zeile 4 angepasst werden. Hier geben Sie als vierten Parameter das erstellte `gradient_bd` an und legen mit dem sechsten Parameter (`true`) fest, dass für den Kopierprozess die Alphadaten von `gradient_bd` herangezogen werden sollen.

Wenn Sie Ihr Dokument nun exportieren, werden Sie sehen, dass auf der Spiegelung ein Alphaverlauf liegt.

Jetzt werden Sie noch Vorkehrungen treffen für den Fall, dass sich in dem Quell-Movieclip Bewegtbilder befinden.

Zunächst müssen Sie sich beim `ENTER_FRAME`-Event anmelden. Fügen Sie folgende if-Anweisung ans Ende des Konstruktors ein:

Tipp

Ausführliche Informationen über die Zeichnen-Engine von Flash finden Sie in der Flash-Hilfe.

Schritt 15

Alphaverlauf anwenden



Zwischenergebnis:
14_WEB2.0-REFLEXION • REFLECTION_
STEP15.AS



▲ Abbildung 7

Die Web 2.0-Spiegelung ist fertig.

Schritt 16

Videoaktualisierung hinzufügen

Zwischenergebnis:
14_WEB2.0-REFLE-
XION • REFLECTION_
STEP16.AS



```
if(video){  
    img.addEventListener(Event.ENTER_FRAME, onImgEnterFrame);  
}
```

Wenn im Konstruktor die `video`-Variable als `true` übergeben wurde (siehe Zeile 22 in Schritt 5), meldet sich das Skript beim `ENTER_FRAME`-Event an. Von nun an wird in jedem Frame die `onImgEnterFrame`-Methode in Ihrem Skript aufgerufen.

Diese Methode erstellen Sie nun. Ergänzen Sie Ihr Skript unter dem Konstruktor um folgende Funktion:

```
1: private function onImgEnterFrame(evt:Event):void  
2: {  
3:     original_bd.draw(img);  
4:     rotate_bd.draw(original_bd, rotate_matrix);  
5:     output_bd.fillRect(output_bd.rect, 0x00000000);  
6:     output_bd.copyPixels(rotate_bd, rotate_bd.rect, new  
7:         Point(0, 0), gradient_bd, new Point(0, 0), true);  
8: }
```

Die Zeilen 3 und 4 verhalten sich exakt wie die Zeilen 33 und 38 im Konstruktor (siehe Beispieldatei auf der Buch-DVD).

Sie setzen den Bildinhalt von `output_bd` in Zeile 5 zurück, indem sie die Fläche mit der Farbe `0x00000000` füllen.

Zeile 6 verhält sich genau wie Zeile 37 im Konstruktor.

Damit das korrekte Verhalten getestet werden kann, sollte sich im Quell-Movieclip natürlich auch animierter Inhalt befinden. Im Beispiel auf der DVD bewegt sich, zu Testzwecken, einfach ein Kreis innerhalb des Bildes.



▲ **Abbildung 8**

Auch animierte Inhalte können jetzt gespiegelt werden.

Flash-Physics und 3D-Programmierung

Schneefall simulieren	152
Springende Bälle	163
Kabelschwingung.....	175
3D-Programmierung in Flash CS3	184
3D-Sternenfeuer	188
3D-Bildergalerie.....	198
3D-Karussellmenü	218

Schneefall simulieren

Fallenden Schnee im Wind simulieren

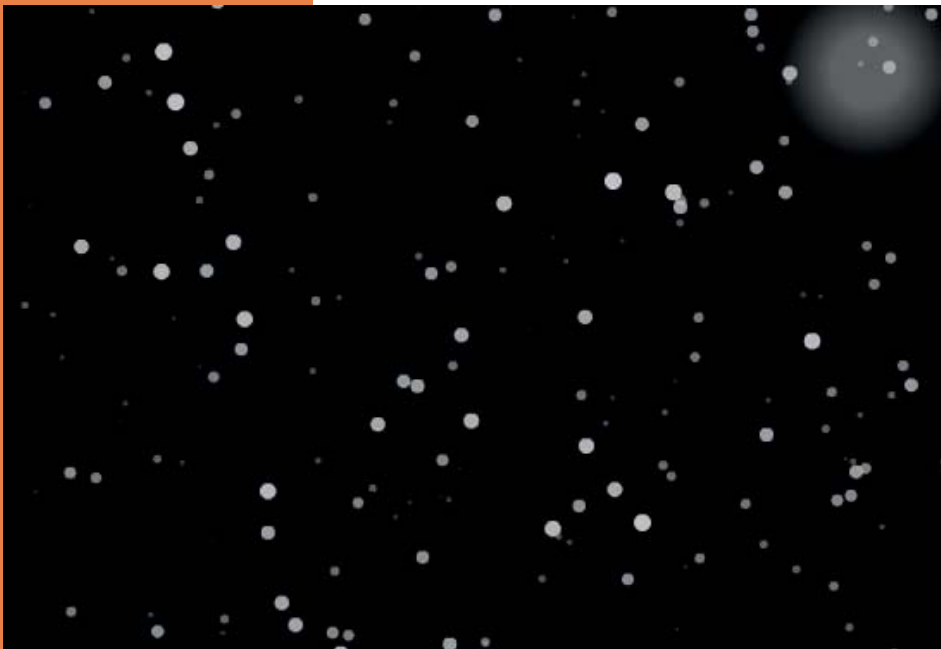
```
class Schneeflocke extends MovieClip {
    private var speedY:Number;
    private var speedX:Number;
    private var peak:Number;
    private var posX:Number;
    function Schneeflocke()
    {
        speedX = Math.random()*10+5;
        peak = Math.random()*1;
    }
    public function set speed(value:Number):Void
    {
        this.speedY = value;
    }
    private function onEnterFrame():Void
    {
```

Mit Hilfe von ActionScript ist es ein Leichtes, Objekten auf der Bühne Leben einzuhauchen – sie zu bewegen, zu verändern und in Interaktion mit dem Benutzer treten zu lassen. In diesem Workshop lernen Sie, wie Sie mit ActionScript Schneefall programmieren können. Der Schnee wird vom Wind gesteuert, den Sie mit der Maus in verschiedene Richtungen lenken können.

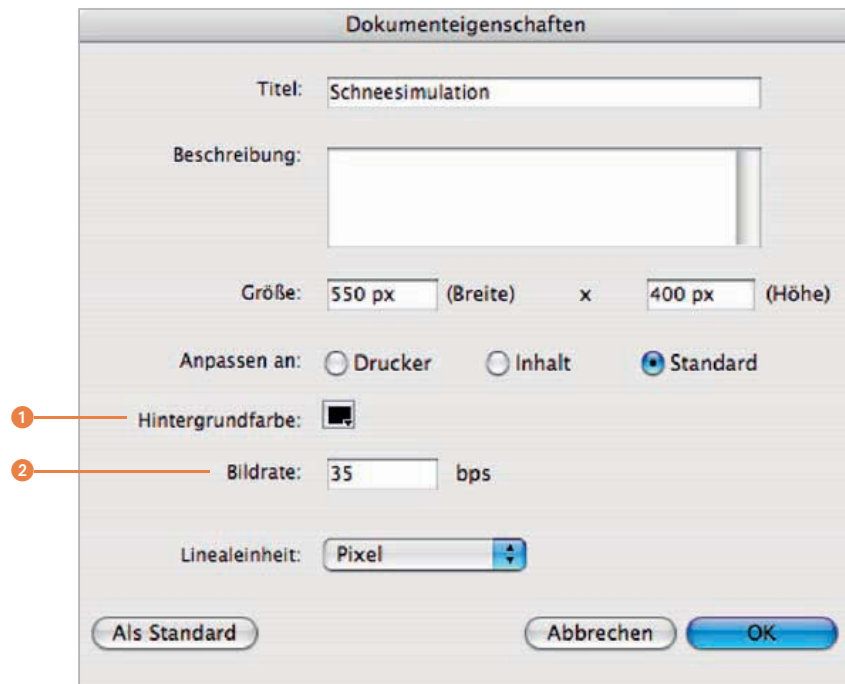
Zielsetzungen:

- Schneeflocke erzeugen
- Schneefall programmieren
- Wind mit der Maus erzeugen

[Ergebnis: 15_SCHNEEFALL-SIMULIEREN • SCHNEESIMULATION.SWF]



Erstellen Sie ein neues AS2-Flash-Dokument, und speichern Sie es im Projektordner SCHNEESIMULATION unter dem Namen SCHNEESIMULATION.FLA ab. Stellen Sie die BILDRATE auf 35 bps **2**. Eine höhere Bildrate ist notwendig, um flüssige Animationen erzeugen zu können (da wir die Werte unserer Schneeflocken in jedem Bild bzw. Frame neu berechnen werden). Stellen Sie die HINTERGRUNDFARBE **1** auf Schwarz.



Schritt 1 Filmeinstellungen vornehmen

▲ **Abbildung 1**
Dokumenteigenschaften
über MODIFIZIEREN • DOKU-
MENT festlegen

Als Schneeflocke reicht in diesem Fall ein einfacher gefüllter Kreis. Wählen Sie hierzu in der Werkzeugpalette das Ellipsenwerkzeug , und stellen Sie die Füllfarbe **4** auf 0xFFFFFF. Die Strichfarbe **3** wird deaktiviert.

Auf der Bühne zeichnen Sie nun mit gehaltener -Taste einen kleinen Kreis. Danach stellen Sie im Eigenschaftenfenster die Breite und Höhe des Kreises auf jeweils 10 Pixel.

Markieren Sie den Kreis, und konvertieren Sie ihn mit **F8** in ein Symbol vom Typ MOVIECLIP. Geben Sie diesem den NAMEN »flocke«, und positionieren Sie den REGISTRIERUNGSPUNKT **5** mittig (siehe Abbildung 3).

Danach aktivieren Sie die Checkbox EXPORT FÜR ACTIONSCRIPT **7**, um dynamisch Instanzen des Movieclips auf der Bühne erzeugen zu können. Der BEZEICHNER steht automatisch auf »flocke«. Tragen Sie als KLASSE »Schneeflocke« **6** ein. In dieser Klasse werden Sie die Funktionalität der Schneeflocke realisieren.

Schritt 2 Schneeflocken-Movieclip erzeugen



▲ **Abbildung 2**
Ellipsenwerkzeug einstellen

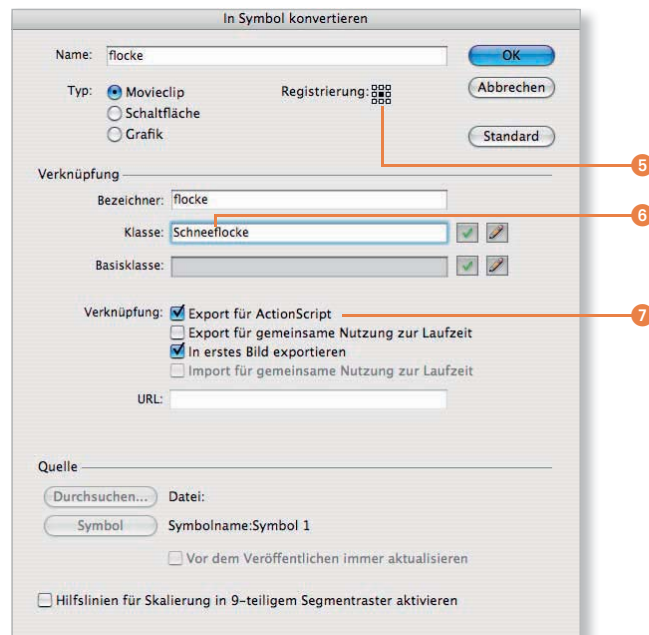


Abbildung 3 ►
Einstellungen für den
Flocken-Movieclip

Nach der Bestätigung erscheint ein Movieclip in der Bibliothek. Löschen Sie nun den flocke-Movieclip von der Bühne, da Sie die Flocken dynamisch erstellen werden.

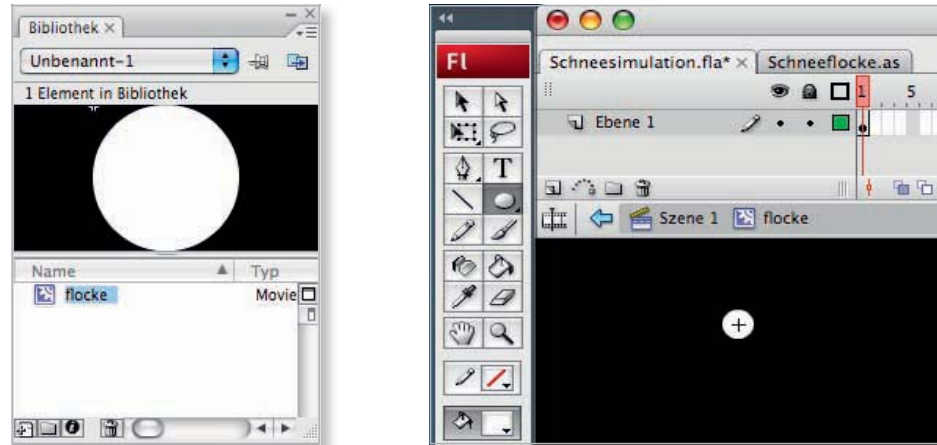


Abbildung 4 ►
Die Schneeflocke wurde
angelegt und erscheint auf
der Bühne (rechts) und in
der Bibliothek (links).

Schritt 3 **Schneeflocken auf der** **Bühne erzeugen**

Klicken Sie in das erste Bild der ersten Ebene in der Hauptzeitleiste, und öffnen Sie das ActionScript-Fenster mit **[F9]** bzw. **[Alt]+[F9]**.

Zunächst deklarieren Sie eine Variable, in der die Anzahl der Schneeflocken gespeichert wird.

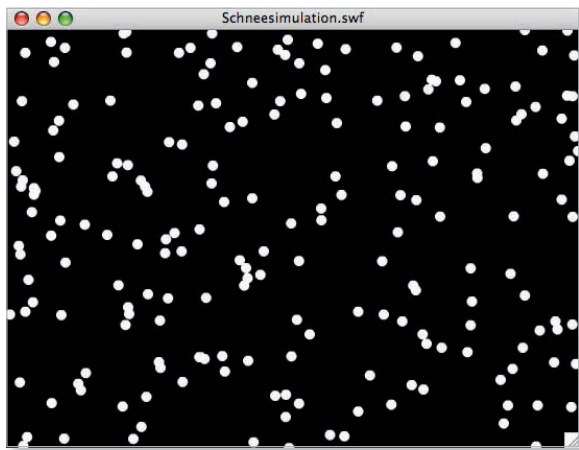
```
1: var numFlocken:Number = 200;
```

Später können Sie diesen Wert beliebig anpassen. Als Nächstes erzeugen Sie innerhalb einer `for`-Schleife Bühneninstanzen des `flocke-Movieclips` per `attachMovie()`.

Um die gesamte Bühne mit Flocken an zufälligen Positionen zu füllen, verwenden Sie die `Math.random`-Methode, die einen Fließkommawert zwischen 0 und 1 zurückliefert. Diesen multiplizieren Sie jeweils mit der Bühnenbreite und Bühnenhöhe und erhalten somit einen Wert, der zwischen Null und der Breite bzw. der Höhe der Bühne liegt. Den so errechneten Wert weisen Sie der `x`- und `y`-Eigenschaft der Flocke zu:

```
2: for (var i:Number = 0; i < numFlocken; i++){
3:     var xPos:Number = Math.random()*Stage.width;
4:     var yPos:Number = Math.random()*Stage.height;
5:     var flocke = this.attachMovie("flocke","flocke_"+i,i);
6:     flocke._x = xPos;
7:     flocke._y = yPos;
8: }
```

Wenn Sie nun den Film testen (`Strg`/`⌘` + `↵`), sehen Sie Ihre Schneeflocken überall auf der Bühne verteilt.



Um einen schöneren Effekt zu bekommen, werden wir nun die Schneeflocken in der Tiefe des Raums verteilen. Dafür nutzen Sie folgenden Trick:

Erweitern Sie die `for`-Schleife ganz unten um die Zeile:

```
var scale:Number = Math.random()*90+10;
```

Achtung

Das Ausgabefenster zeigt die Fehlermeldung, dass die Klasse oder Benutzeroberfläche »Schneeflocke« nicht geladen werden konnte. Diese Meldung erscheint, weil Sie zwar eine Klassenverknüpfung für den `flocke-Movieclip` angegeben haben, jedoch noch keine Klasse bzw. noch keine Datei `SCHNEEFLOCKE.AS` erstellt haben. Die Meldung sollte Sie nicht weiter beunruhigen, denn der Film funktioniert trotzdem.

Schritt 4 Film testen

◀ Abbildung 5

Die Schneeflocken werden dynamisch und an einer Zufallsposition erstellt.

Schritt 5 Anpassen der Flockeneigenschaften

In der Variablen `scale` wird ein zufälliger Wert zwischen 10 und 100 gespeichert. Diesen wenden Sie nun auf die `_xscale`- und `_yscale`- sowie auf die `_alpha`-Eigenschaft der Flocken an.

Hierbei gilt eine einfache Regel: Je größer die Entfernung, desto kleiner werden die Schneeflocken. Fügen Sie deshalb direkt unter die eben eingefügte Zeile folgenden Code ein:

```
flocke._xscale = flocke._yscale = flocke._alpha = scale;
```

Schritt 6

Film erneut testen

Testen Sie nun den Film, und schauen Sie sich das Ergebnis an. Wie Sie sehen, scheinen die Schneeflocken im Raum verteilt zu sein.



Abbildung 6 ►

Durch die Verkleinerung einiger Schneeflocken entsteht ein Eindruck von Tiefe.

Schritt 7

Eigenschaften der Schneeflocken erweitern

Um den Effekt noch realistischer zu machen, passen Sie nun noch den Wert der Fallgeschwindigkeit an die Entfernung an. Auch hier gilt: Je weiter weg, desto langsamer fällt etwas, zumindest scheinbar. Den Wert für die Geschwindigkeitseigenschaft `speed` der Flocke werden Sie schon auf der Hauptzeitleiste setzen, jedoch erst später in der Schneeflocken-Klasse implementieren.

Erweitern Sie die `for`-Schleife um folgende zwei Zeilen:

```
var speed:Number = scale/20;  
flocke.speed = speed;
```

Damit ist das Initialisierungsskript vollständig. Es sorgt dafür, dass die Schneeflocken auf einer zufälligen Position auf der Bühne erscheinen und dass sie verschieden groß und durchsichtig sind.

Legen Sie über `[Strg]/[⌘]+[N]` eine neue ACTIONSCRIPT-DATEI an, und speichern Sie diese als SCHNEEFLOCKE.AS im Projektordner ab.

Öffnen Sie die Datei, und schreiben Sie zunächst den Klassenrumpf mit Konstruktor:

```
1: class Schneeflocke extends MovieClip {  
2:     function Schneeflocke()  
3:     {  
4:     }  
5: }
```

Ihre Klasse erbt von `MovieClip`, da Sie diese dem `flocke-Movieclip` in der Bibliothek zugeordnet haben. So können Sie in Ihrer Klasse auf die `Movieclip-Eigenschaften` zugreifen.

Als Erstes deklarieren Sie eine Variable `speedY`, in der später durch eine Setter-Methode `speed` der Wert für die Fallgeschwindigkeit gespeichert wird. Sie erinnern sich: Auf der Hauptzeitleiste haben Sie bereits den Aufruf `flocke.speed = speed` für eine Setter-Methode durchgeführt. Nehmen Sie die Deklaration direkt über dem Konstruktor vor:

```
private var speedY:Number;
```

Jetzt fehlt noch die Setter-Methode, um die Variable mit einem Wert von außen belegen zu können. Ergänzen Sie folgenden Code unter dem Konstruktor:

```
public function set speed(value:Number):Void  
{  
    this.speedY = value;  
}
```

Um die `y-Position` in jedem Bild pro Sekunde neu zu berechnen, schreiben Sie die Methode `onEnterFrame()`, die automatisch in jedem Frame ausgeführt wird.

Fügen Sie sie unter die eben eingefügte `speed-Methode` ein.

Schritt 8

Die Klasse »Schneeflocke«

Schritt 9

Die Fallgeschwindigkeit setzen

Schritt 10

Die Schneeflocken fallen lassen

```
private function onEnterFrame():Void
{
    this._y += speedY;
}
```

Die y-Position der Flocke wird damit in jedem Bild um `speedY` vergrößert.

Schritt 11

Film testen

Testen Sie nun den Film wieder. Wie Sie sehen, bewegen sich die Flocken mit unterschiedlicher Geschwindigkeit Richtung »Boden« bzw. unterer Bühnenrand. Jedoch verschwinden die Flocken einfach unter der Bühne, und es kommen auch keine Flocken mehr nach.

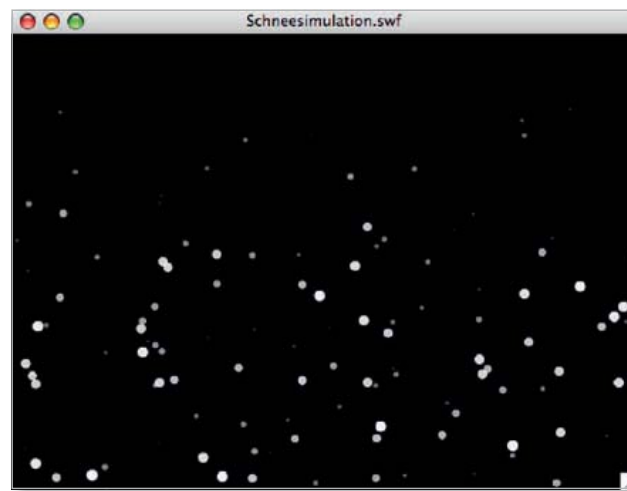


Abbildung 7 ►

Die Schneeflocken fallen, allerdings im Moment nur für kurze Zeit.

Schritt 12

Ewiger Schnee

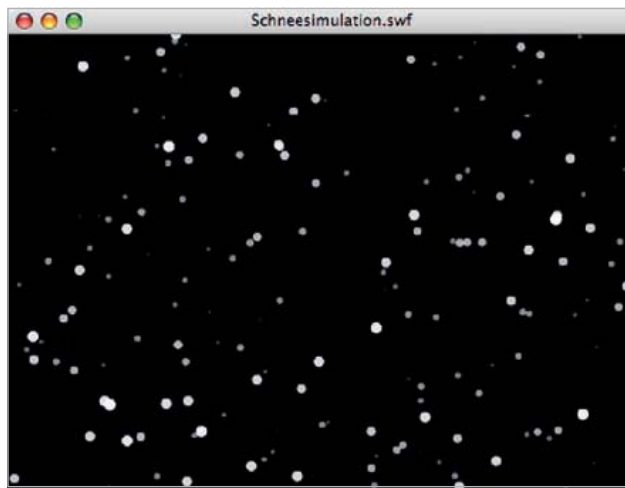
Damit es kontinuierlich weiterschneit, werden Sie nun eine Methode `checkBottom()` programmieren, in der abgefragt wird, ob sich die Schneeflocke außerhalb des unteren Bühnenrands befindet. Sollte dies der Fall sein, wird die Schneeflocke wieder am oberen Rand positioniert. Ergänzen Sie diese Methode unter der Methode `onEnterFrame()` aus Schritt 10:

```
private function checkBottom():Void
{
    if(this._y > Stage.height){
        this._y = 0;
    }
}
```

Damit diese Methode auch ihren Dienst erfüllt, rufen Sie `checkBottom()` in der letzten Zeile der `onEnterFrame`-Methode auf. Eine einfache Abfrage mit viel Wirkung!

```
checkBottom();
```

Testen Sie nun Ihren Film, und bestaunen Sie das Ergebnis. Herzlichen Glückwunsch, Sie haben Ihren ersten Schneefall programmiert!



Das Ergebnis kann sich zwar bereits sehen lassen, doch wir wollen noch ein wenig mehr Dynamik und Natürlichkeit. Das nächste Ziel ist, mit einer Sinusfunktion die Schneeflocken auf der x-Achse hin und her zu bewegen. So entsteht der Eindruck, die leichten Flocken würden im Wind schaukeln.

Deklarieren Sie zu diesem Zweck zwei weitere Variablen im Deklarationsbereich der Schneeflocke-Klasse:

```
private var speedX:Number;  
private var peak:Number;
```

In `speedX` wird ein Wert gespeichert, der definiert, wie schnell sich die Flocke von einer Seite zur anderen bewegt. `peak` bestimmt den Ausschlag der gleich zu errechnenden Sinuskurve und somit den Bereich auf der x-Achse, in dem die Schneeflocke von rechts nach links und umgekehrt bewegt wird. Diese beiden Variablen sollen durch Zufallswerte bestimmt werden.

Erweitern Sie hierzu den Konstruktor folgendermaßen:

Schritt 13

Film testen

◀ Abbildung 8

Es schneit – wenn Sie wollen, für immer.

Schritt 14

Flocken im Wind bewegen

```
function Schneeflocke()
{
    speedX = Math.random()*10+5;
    peak = Math.random()*1;
}
```

In der `onEnterFrame`-Methode werden Sie daraus eine Sinuskurve berechnen. Fügen Sie folgende Zeilen ans Ende dieser Methode ein:

```
var timer : Number = getTimer()/100;
this._x += Math.sin(timer/speedX)*peak;
```

Um eine Sinuskurve zu erzeugen, benötigt man einen Wert, der permanent hochgezählt wird und der `Math.sin`-Methode übergeben wird. Diese Methode erzeugt daraus einen sich immer zwischen -1 und 1 verändernden Wert. `getTimer()` liefert die Millisekunden, die seit dem Start des Flash-Films vergangen sind, also einen immer größer werdenden Wert. Da dieser Wert jedoch zu hoch ist und das Resultat eine zu schnelle Bewegung wäre, teilen Sie den Wert durch 100 . Der `Math.sin`-Methode übergeben Sie den `timer`, den Sie durch `speedX` teilen, um ein individuelles Ergebnis entsprechend `speedX` zu erreichen.

Schritt 15

Film testen

Testen Sie den Film. Die Flocken fallen und bewegen sich im Wind.



Abbildung 9 ►

Die Flocken bewegen sich scheinbar im Wind.

Schritt 16

Wind mit der Maus steuern

Sie haben nun im Grunde bereits ein gutes Ergebnis erzeugt. Aber wir wollen noch mehr Dynamik! Ziel ist es, den Schneefall mit der Maus in verschiedene Richtungen lenken zu können. Abhängig von der Mausposi-

tion zur Mitte der Bühne soll der »Wind« stärker und schwächer sein und außerdem die Richtung wechseln.

Deklarieren Sie im Deklarationsbereich der Klasse die Variable `posX`, die dazu dienen wird, die Flockenrichtung auf der x-Achse zu ändern:

```
private var posX:Number;
```

In der `onEnterFrame`-Methode fragen Sie nun die Mausposition im Verhältnis zur Bühnenmitte ab, indem Sie von der Mausposition einfach die Bühnenbreite geteilt durch 2 abziehen. Erweitern Sie die `onEnterFrame`-Methode um folgende Zeilen:

```
1: var mousePosition:Number = (_root._xmouse-Stage.width/2)/50;  
2: posX = -mousePosition;  
3: this._x += posX;
```

Der in Zeile 1 errechnete Wert ist etwas zu hoch und würde die Schneeflocken zur Seite flitzen lassen. Deswegen teilen Sie ihn durch 50. In der nächsten Zeile weisen Sie der Variablen `posX` den errechneten Wert zu. Wie Sie sehen, weisen Sie den negativen Wert von `mousePosition` zu. Dies zielt darauf ab, die Schneeflocken immer entgegen der Mausrichtung fliegen zu lassen, wie Sie gleich beim Testen feststellen werden. Schlussendlich addieren Sie den errechneten Wert zu der x-Position der Flocke.

Testen Sie nun den Film. Die Schneeflocken reagieren auf die Mausposition. Jedoch fliegen die Flocken, je nachdem, wohin Sie die Maus bewegen, über den Rand hinaus und es entsteht eine Lücke.



Schritt 17

Film testen

◀ Abbildung 10

Die Schneeflocken ändern zwar ihre Richtung, es entsteht jedoch eine unschöne Lücke.

Schritt 18

Flocken im Bild behalten

Genau wie in Schritt 12, in dem die y-Position korrigiert wird, benötigen Sie eine Methode, die die x-Position korrigiert. Es gilt abzufragen, ob die x-Position entweder größer als die Bühnenbreite oder kleiner als 0 ist. In anderen Worten: ob die Schneeflocke die Bühne nach rechts oder links verlässt. Die x-Position soll dann entsprechend vom linken zum rechten Rand gesetzt werden und umgekehrt. Die x-Position steuern Sie über die Variable `xPos`. Fügen Sie die `checkSites`-Methode folgendermaßen unter die `checkBottom`-Methode ein:

```
private function checkSites():Void{
    if(this._x >= Stage.width)
        posX -= Stage.width;
    else if(this._x <= 0)
        posX += Stage.width;
}
```

In der `onEnterFrame`-Methode müssen Sie diese Methode nun noch aufrufen, bevor die x-Position angeglichen wird. Ergänzen Sie Zeile 2 innerhalb der `onEnterFrame`-Methode:

```
1:  posX = -mousePosition;
2:  checkSites();
3:  this._x += posX;
3:  }
```

Schritt 19

Ergebnis testen

Testen Sie nun Ihren Film, und spielen Sie ein wenig mit den Schneeflocken im Wind. Wie Sie sehen, verlassen die Flocken nie wirklich das Bild, sondern werden für den Fall, dass sie die Bühnenränder erreicht haben, wieder auf der gegenüberliegenden Seite positioniert.



Abbildung 11 ►
Ein Ergebnis, das sich sehen lassen kann!

Springende Bälle

Physikalische Eigenschaften programmieren

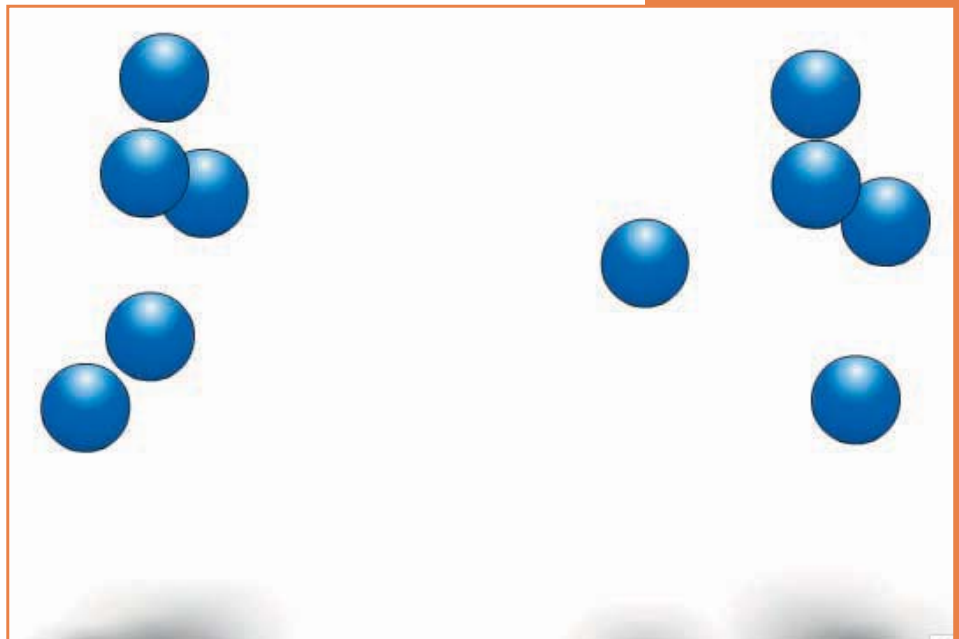
Haben Sie sich schon immer gefragt, wie zum Beispiel in Online-Spielen die Objekte so realistisch bewegt werden? Genau das soll Ihnen dieser Workshop zeigen, in dem Sie die physikalischen Bedingungen der Realität möglichst genau abbilden. Sie werden Bälle erschaffen, die Sie werfen können, die vom Boden, der Decke und den Wänden abprallen und nach einer Weile Hüpfen zum Stillstand kommen.

```
import flash.filters.DropShadowFilter;
class BouncingBall extends MovieClip {
    private var friction:Number = 0.98;
    private var bounce:Number = -0.7;
    private var gravity:Number = 0.8;
    private var velocityX:Number;
    private var velocityY:Number;
    private var top:Number = 0;
    private var left:Number = 0;
    private var bottom:Number = Stage.height;
    private var right:Number = Stage.width;
    private var isDragging:Boolean = false;
    private var lastX:Number;
    private var lastY:Number;
    private var shadow:DropShadowFilter;
    function BouncingBall()
    {
```

Zielsetzungen:

- Ball zeichnen
- Physik-Engine programmieren
- Schatten hinzufügen

[Ergebnis: 16_SPRINGENDE-
BAELLE • BALLSIMULATION.
SWF]



Schritt 1 Vorbereitungen

Im ersten Schritt legen Sie einen neuen Ordner auf Ihrer Festplatte an und nennen diesen BALLSIMULATION. Hier werden Sie die von Ihnen erstellten Dateien zu diesem Workshop abspeichern.

Schritt 2 Dokument erstellen

Erstellen Sie ein neues AS2-Flash-Dokument mit der Größe 550x400 Pixel und einer BILDRATE von 30bps. HINTERGRUNDFARBE und sonstige Einstellungen spielen keine Rolle.

Speichern Sie die Datei unter dem Namen BALLSIMULATION.FLA in dem Projektordner.

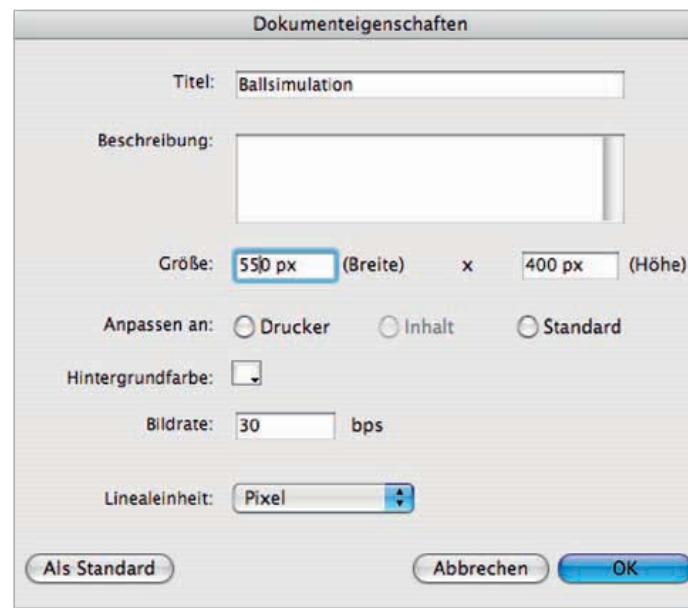


Abbildung 1 ►
Stellen Sie Größe und Bildrate in den Dokumenteigenschaften ein.

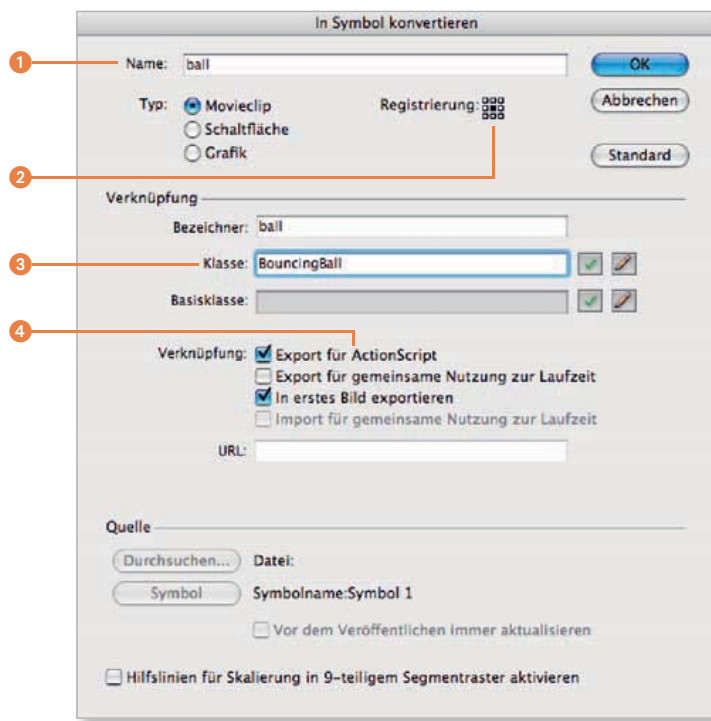
Schritt 3 Ball erstellen



Abbildung 2 ▲
Einstellungen für das Ellipsenwerkzeug

Wählen Sie in der Werkzeugpalette das Ellipsenwerkzeug , und stellen Sie die Füllfarbe auf #0066cc. Die Strichfarbe stellen Sie auf #000000. Auf der Bühne zeichnen Sie nun mit gedrückter -Taste einen kleinen Kreis. Danach stellen Sie im Eigenschaftenfenster die Breite und Höhe des Kreises auf jeweils 50 Pixel.

Markieren Sie den Kreis, und konvertieren Sie ihn mit  in ein Symbol (Movieclip). Geben Sie diesem den NAMEN »ball« , und positionieren Sie den Registrierungspunkt mittig . Dann aktivieren Sie die Checkbox EXPORT FÜR ACTIONSCRIPT , um dynamisch Instanzen des Balls auf der Bühne erzeugen zu können. Der BEZEICHNER steht automatisch auf »ball«. Tragen Sie als KLASSE »BouncingBall«  ein. In dieser Klasse werden Sie die Funktionalität des Balls realisieren.



◀ **Abbildung 3**
Der Ball wird in einen
Movieclip konvertiert.

Nach der Bestätigung erscheint der Movieclip in der Bibliothek. Löschen Sie nun den ball-Movieclip von der Bühne, da Sie diesen dynamisch auf der Bühne erscheinen lassen werden.

Wählen Sie **DATEI • NEU • ACTIONSCRIPT-DATEI**. Speichern Sie die Datei unter dem Namen **BOUNCINGBALL.AS** im Projektordner.

Schritt 4 ActionScript-Datei erzeugen

Als Erstes programmieren Sie die Klassen-Definition inklusive des Konstruktors. Übernehmen Sie dazu folgende Zeilen:

Schritt 5 Die BouncingBall-Klasse

```
class BouncingBall extends MovieClip {
    function BouncingBall()
    {
    }
}
```

Da Ihr ball-Movieclip mit der BouncingBall-Klasse verknüpft ist und Sie in dieser Klasse auf Movieclip-Eigenschaften zugreifen werden, erbt sie von **MovieClip**. Die Vererbung geben Sie über **extends** an.

Ein kleiner Exkurs über physikalische Eigenschaften

Wenn Sie eine Bewegung möglichst nah an der Realität programmieren möchten, müssen Sie die Bedingungen der Bewegung auf ein Minimum reduzieren und dann in ihre Einzelteile aufteilen. Für eine Bewegung benötigen Sie in erster Linie eine Beschleunigung, einen **Antrieb**. In ActionScript übersetzt, ist dies ein flexibler Wert, der die `_x`- bzw. `_y`-Eigenschaft des Movieclips verändert. Dann können Sie davon ausgehen, wenn Sie nicht gerade ein Weltraum-Spiel programmieren, dass **Gravitation** vorhanden ist. Das bedeutet eine konstante Kraft, die permanent auf einen Gegenstand so einwirkt, dass dieser in Richtung Boden befördert wird. Übersetzt in ActionScript: Sie haben einen konstanten Wert, der permanent, also in jedem Frame, auf die `_y`-Eigenschaft addiert wird. Dazu kommt die **Reibung**. Reibung ist immer vorhanden – in der Luft, am Boden, im Wasser. Sie unterscheidet sich nur durch die Beschaffenheit des Objekts im Verhältnis zu der umgebenden Materie. Reibung führt zum Beispiel dazu, dass ein Ball, der ins Rollen gebracht wird, nach einer gewissen Zeit wieder zum Stillstand kommt. Reibung ist also ein Wert, der die Bewegung bzw. die Beschleunigung vermindert. Für unser Beispiel fehlt nur noch eine Komponente: die **Federkraft**. Diese bestimmt die Art der Beschleunigung beim Auftreffen auf ein Hindernis. Man könnte noch weitergehen und zum Beispiel die Masse der Gegenstände mit einbeziehen. Das jedoch können Sie sich für eigene Überlegungen, in schlaflosen Nächten voller Euphorie eine kleine Physikengine programmiert zu haben, aufheben.

Schritt 6

Deklaration der Variablen

Deklarieren Sie nun unterhalb der Klassen-Definition – also direkt über dem Konstruktor – folgende Klassenvariablen:

```
1: private var friction:Number = 0.98;
2: private var bounce:Number = -0.7;
3: private var gravity:Number = 0.8;
4: private var velocityX:Number;
5: private var velocityY:Number;
```

In Zeile 1 legen Sie in `friction` den Wert der Reibung fest. In der nächsten Zeile bestimmen Sie die Federkraft `bounce`. Danach legen Sie den Gravitationswert `gravity` fest. In den Zeilen 4 und 5 deklarieren Sie `velocityX` und `velocityY`: Variablen für die Beschleunigungswerte.

Schritt 7

Initialisieren der Anfangsbeschleunigung

Legen Sie nun im Konstruktor mit den folgenden Zeilen die Anfangsbeschleunigung fest. Diese Beschleunigung soll der Ball haben, wenn der Flashfilm gestartet wird:

```
velocityX = Math.random() * 40-20;  
velocityY = Math.random() * -20-10;
```

In der ersten Zeile weisen Sie `velocityX` einen zufälligen Wert zwischen 20 und –20 zu. `velocityY` weisen Sie einen zufälligen Wert zwischen –10 und –30 zu. Diese Werte bestimmen die anfängliche Beschleunigung auf der x- und y-Achse.

In diesem Schritt werden Sie mit den bisher deklarierten und initialisierten Variablen die x- und y-Position des Balls in der Methode `onEnterFrame()` verändern. Schreiben Sie unterhalb des Konstruktors folgenden Code:

```
1: private function onEnterFrame():Void  
2: {  
3:     velocityY += gravity;  
4:     velocityX *= friction;  
5:     velocityY *= friction;  
6:     this._x += velocityX;  
7:     this._y += velocityY;  
8: }
```

In Zeile 3 addieren Sie `gravity` auf `velocityY`, die y-Beschleunigung. In den beiden nächsten Zeilen multiplizieren Sie die x- und y-Beschleunigung mit der Reibung `friction`. Da der `friction`-Wert unter 1 liegt, wird die Beschleunigung Schritt für Schritt reduziert. In den Zeilen 6 und 7 addieren Sie die Beschleunigungswerte `velocityX` und `velocityY` schließlich auf die `_x`- und `_y`-Eigenschaft des Balls.

Wechseln Sie nun wieder zu Ihrem Flash-Dokument. Klicken Sie auf das erste Bild der ersten Ebene in der Hauptzeitleiste, und öffnen Sie mit `[F9]` bzw. `[Alt]+[F9]` das ActionScript-Fenster. Fügen Sie folgenden Code ein:

```
1: var ball:MovieClip = attachMovie("ball", "ball", 0);  
2: ball._x = Stage.width / 2;  
3: ball._y = Stage.height / 2;
```

Sie erzeugen mit `attachMovie()` eine Instanz des Balls auf der Bühne. Danach positionieren Sie den Ball mittig auf der Bühne.

Schritt 8

onEnterFrame-Methode programmieren

Schritt 9

Erzeugen des Balls auf der Bühne

Schritt 10

Film testen

Testen Sie nun Ihren Flash-Film mit `Strg`/`⌘` + `↵`. Wie Sie sehen, wird der Ball von der Mitte der Bühne in eine zufällige Richtung geworfen und fällt dann in Richtung Boden. Testen Sie den Film mehrmals hintereinander, indem Sie einfach wieder `Strg`/`⌘` + `↵` betätigen. Wie Sie sehen, wird der Ball immer mit anderer Beschleunigung in immer wieder verschiedene Richtungen geworfen.

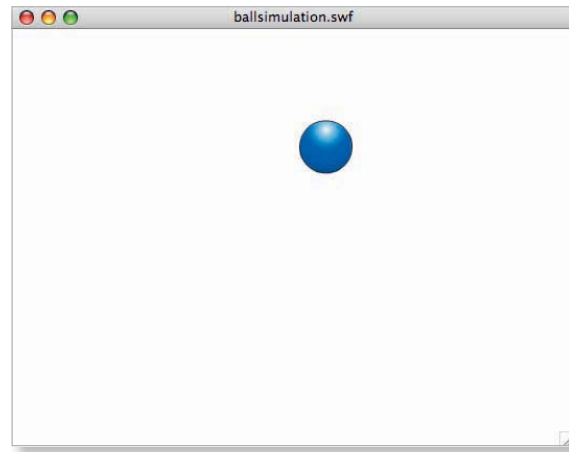


Abbildung 4 ►

Der Ball wird am Anfang der Animation auf die Bühne geworfen.

Schritt 11

Boden, Decke und Wände definieren

Wie Sie beim Ausführen des Flash-Films gesehen haben, fällt der Ball in Richtung Boden. Jedoch fällt er endlos weiter. Der Ball soll allerdings wieder vom Boden abprallen. Schön wäre es, wenn sich solche selbstverständlichen Dinge einfach von selbst lösen würden. Aber das tun sie leider nicht. Also müssen Sie selbst Hand anlegen.

Als Erstes müssen Sie Variablen für alle vier Begrenzungen festlegen: für den Boden, die Decke und die zwei Wände. Deklarieren Sie dafür oben im Deklarationsbereich der Klasse folgende Klassenvariablen:

```
1: private var top:Number = 0;  
2: private var left:Number = 0;  
3: private var bottom:Number = Stage.height;  
4: private var right:Number = Stage.width;
```

In Zeile 1 legen Sie in der Variablen `top` die minimale y-Position fest, die der Ball erreichen darf: 0 Pixel. Der Ball darf damit nicht über die Bühnenhöhe verschwinden. In der nächsten Zeile bestimmen Sie mit der Klassenvariablen `left` die minimale x-Position des Balls: 0 Pixel. Der Ball darf nicht weiter nach links als bis zum Bühnenrand. Zeile 3 bestimmt mit der Variablen `bottom` die maximale y-Position des Balls. Diesen Wert setzen Sie

auf die Höhe der Bühne. Also den untersten Rand der Bühne. In Zeile 4 legen Sie den Wert für die maximale x-Position des Balls in der Variablen `right` fest. Diese erhält den Wert der Bühnenbreite, also des rechten Rands der Bühne.

Es muss permanent geprüft werden, ob der Ball die festgelegte Begrenzung über- oder unterschreitet, und dann entsprechend reagiert werden. Fügen Sie dem Code zu der `onEnterFrame`-Methode folgende Codezeilen hinzu:

```
if (this._x + this._width / 2 > right){
    this._x = right - this._width / 2;
    velocityX *= bounce;
}
else if (this._x - this._width / 2 < left){
    this._x = left + this._width / 2;
    velocityX *= bounce;
}
if (this._y + this._height / 2 > bottom){
    this._y = bottom - this._height / 2;
    velocityY *= bounce;
}
else if (this._y - this._height / 2 < top){
    this._y = top + this._height / 2;
    velocityY *= bounce;
}
```

Mit diesem Code fragen Sie ab, ob die x- oder y-Position des Balls größer oder kleiner ist als die in Schritt 11 festgelegten Begrenzungswerte. Sollte dies der Fall sein, multiplizieren Sie `velocity` mit `bounce`. Da `bounce` ein negativer Wert ist, wird `velocity` ins Gegenteil umgekehrt und somit die Bewegung in die andere Richtung umgelenkt. Da `bounce` dazu noch größer als `-1`, nämlich `-0.7` ist, wird zusätzlich die Beschleunigung ein wenig verringert. Dies entspricht einer zusätzlichen Reibung, die eintritt, wenn der Ball auf ein Hindernis trifft. Da der Registrierungspunkt des Balls in dessen Mitte liegt, müssen Sie bei der Berechnung den Radius des Balls (`_width/2` bzw. `_height/2`) mit einbeziehen.

Testen Sie nun Ihren Film mit `Strg`/`⌘`+`↩`. Wie Sie sehen, fliegt und fällt der Ball wie gewohnt. Allerdings prallt der Ball nun auch an den von Ihnen definierten Rändern ab und ändert die Richtung.

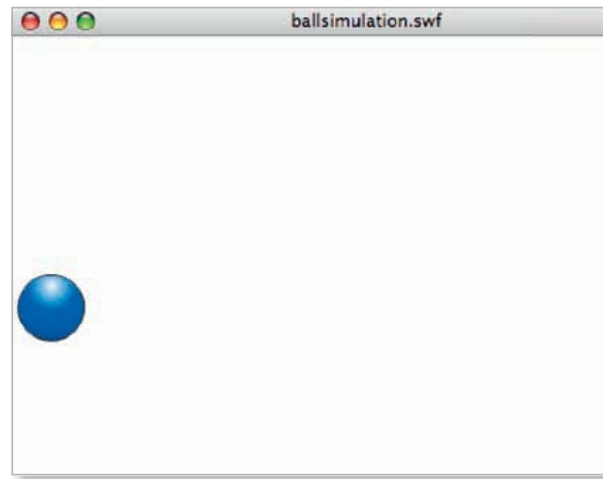
Schritt 12

Ballposition berechnen

Schritt 13

Film testen

Abbildung 5 ►
Der Ball prallt von den Bühnenrändern ab, so wie man es erwarten würde.



Schritt 14

Ball werfen

Boolean

Eine boolesche Variable kann zwei Werte annehmen: `true` oder `false`. Sie dient also dazu, einen Status zu speichern: wahr oder nicht wahr. Eine solche Variable wird demzufolge oft als eine Art Schalter verwendet, um je nach Zustand verschiedenen Code auszuführen.

Schritt 15

Mouse-Events hinzufügen

Sie haben bereits alles programmiert, damit sich der Ball physikalisch korrekt verhält. Jedoch ist der ganze Spaß vorbei, wenn der Ball zum Stillstand kommt. Das reicht uns natürlich nicht. Sie werden die Klasse nun so erweitern, dass Sie den Ball mit der Maus aufheben und werfen können.

Um später in der `onEnterFrame`-Methode festzustellen, ob der Ball angeklickt und verschoben wird, benötigen Sie eine boolesche Variable, die Sie bei einem Klick auf `true` und beim Loslassen auf `false` setzen. Deklarieren Sie die Variable `isDragging` im Deklarationsbereich:

```
private var isDragging:Boolean = false;
```

Sie weisen `isDragging` gleich `false` zu, da Sie davon ausgehen können, dass beim Start des Flash-Films der Ball noch nicht angeklickt ist.

Nun müssen Sie das »Aufheben« und »Loslassen« des Balls abfangen. Dazu verwenden Sie die Methoden `onPress()`, `onRelease()` und `onReleaseOutside()`. Erweitern Sie Ihren Quelltext unterhalb der Methode `onEnterFrame()` aus Schritt 8 um folgende Codezeilen:

```
private function onPress():Void
{
    isDragging = true;
    startDrag(false);
}
private function onRelease():Void
{
```

```

        isDragging = false;
        stopDrag();
    }
    private function onReleaseOutside():Void
    {
        isDragging = false;
        stopDrag();
    }

```

In der `onPress`-Methode setzen Sie `isDragging` auf `true`. Der Aufruf von `startDrag(false)` hat zur Folge, dass der angeklickte Ball nun der Maus folgt. In der `onRelease`-Methode setzen Sie `isDragging` wieder auf `false` und rufen `stopDrag()` auf, womit der Ball wieder losgelassen wird. Diesen Code übernehmen Sie ebenfalls für die `onReleaseOutside`-Methode. Dies dient nur der Sicherheit, da es vorkommen kann, dass die Position des Balls beim Verschieben nicht so schnell berechnet wird, wie Sie die Maus bewegen. So lassen Sie unter Umständen die Maustaste los, wenn sich der Mauszeiger gar nicht über dem Ball befindet. Der Ball würde dann weiterhin Ihrer Maus folgen.

Da in der `onEnterFrame`-Methode in jedem Frame die Position des Balls immer neu berechnet und gesetzt wird, egal ob dieser von der Maus bewegt wird, müssen Sie nun den gesamten Code in eine `if`-Abfrage legen, die diesen nur dann ausführt, wenn `isDragging` auf `false` gesetzt ist.

Schreiben Sie vor die erste Anweisung in der `onEnterFrame`-Methode folgende Codezeile:

```
if(!isDragging){
```

Vergessen Sie nicht, die `if`-Abfrage am Ende des Codes, bevor die `onEnterFrame`-Methode geschlossen wird, durch eine geschweifte Klammer `()` wieder zu schließen.

Testen Sie nun wieder Ihren Flash-Film. Wie Sie sehen, springt der Ball wie gewohnt in eine Richtung. Wenn Sie nun den Ball mit der Maus anklicken, können Sie ihn bewegen.

Wenn Sie den Ball wieder loslassen, fällt er zu Boden. Aber von Werfen kann noch nicht die Rede sein. Dazu müssen Sie nur noch wenige Zeilen programmieren.

Schritt 16

`onEnterFrame`-Methode anpassen

Schritt 17

Film testen

Schritt 18

Variablen deklarieren

Um den Ball zu werfen, können Sie wie folgt vorgehen: Sie speichern die x- und y-Position des Balls in dem Moment ab, wo auf den Ball geklickt wird. Beim Loslassen errechnen Sie die Differenz aus diesen Werten und der neuen Position. Diese Differenz setzen Sie als neue Beschleunigung ein. Das bedeutet kurz gesagt: Je mehr Sie ausholen, desto kräftiger der Wurf.

Deklarieren Sie zu diesem Zweck zwei Klassenvariablen im Deklarationsbereich:

```
private var lastX:Number;
private var lastY:Number;
```

Schritt 19

Errechnen der Wurfkraft

Weisen Sie nun in der `onPress`-Methode den neu deklarierten Variablen die momentane x- bzw. y-Position zu. Verändern Sie die `onPress`-Methode in Ihrer Klasse wie folgt:

```
1: private function onPress():Void
2: {
3:     isDragging = true;
4:     startDrag(false);
5:     lastX = this._x;
6:     lastY = this._y;
7: }
```

In den Zeilen 5 und 6 speichern Sie die x- und y-Werte.

Schritt 20

Anpassen der `onEnterFrame`-Methode

Nun ist noch eine Sache zu tun. In der `onEnterFrame`-Methode haben Sie im Schritt 16 eine `if(!isDragging)`-Bedingung für die Berechnung der Position des Balls eingeführt. Schreiben Sie nun an das Ende dieser `if`-Bedingung den `else`-Rumpf mit folgendem Code:

```
else{
    velocityX = (this._x - lastX)/2;
    velocityY = (this._y - lastY)/2;
}
```

Wenn also der Ball wieder losgelassen wird, weisen Sie `velocityX` und `velocityY` die Differenz aus den momentanen Koordinaten und den in `lastX` und `lastY` gespeicherten Werten zu. Da dies jedoch etwas viel Beschleunigung wäre, teilen Sie das Ergebnis noch durch 2.



▲ Abbildung 6

Der Ball kann nun geworfen werden.

Testen Sie jetzt wieder Ihren Flash-Film mit `Strg`/`⌘`+`↵`. Heben Sie den Ball mit der Maus auf und werfen Sie ihn, indem Sie die Maus bewegen und dann die Maustaste loslassen. Herzlichen Glückwunsch, Sie haben ganze Arbeit geleistet!

Um die ganze Sache noch ein wenig zu verschönern, werden Sie dem Ball nun einen Schatten hinzufügen. Importieren Sie zu diesem Zweck ganz oben in der ersten Zeile in Ihrem Skript die `DropShadowFilter`-Klasse:

```
import flash.filters.DropShadowFilter;
```

Deklarieren Sie anschließend eine Klassenvariable `shadow`:

```
private var shadow:DropShadowFilter;
```

Fügen Sie zum Initialisieren von `shadow` dem Konstruktor der `Bouncing-Ball`-Klasse folgende Zeilen hinzu:

```
shadow = new DropShadowFilter();  
shadow.angle = 90;  
shadow.quality = 3;
```

In der ersten Zeile legen Sie den Einfallswinkel des Schattens auf 90 Grad fest. Die Qualität setzen Sie auf das Maximum 3.

Fügen Sie nun gleich am Anfang der `onEnterFrame`-Methode noch vor der `if`-Schleife folgende Codezeilen hinzu:

```
1: shadow.distance = bottom-this._y;  
2: shadow.blurX = shadow.blurY=shadow.distance/5;  
3: shadow.alpha = this._y/bottom;  
4: this.filters = [shadow];
```

Die Distanz des Schattens errechnen Sie in Zeile 1 durch die Differenz von `bottom` und der `y`-Position. Somit wird der Schatten immer am Boden bleiben. Wie stark der Blur-Effekt des Schattens ist, errechnen Sie wiederum aus der Distanz des Schattens: Je höher, desto mehr Blur. In Zeile 3 errechnen Sie den `alpha`-Wert des Schattens ganz einfach dadurch, dass Sie die `_y`-Eigenschaft des Balls durch `bottom` teilen. Je höher der Ball fliegt, desto niedriger ist der Wert. In Richtung Boden nähert sich der Wert eins. In

Schritt 21

Film testen

Schritt 22

Schatten hinzufügen

Erweiterungsmöglichkeiten

Überlegen Sie, wie Sie die Klasse erweitern können. Sie können zum Beispiel Getter- und Setter-Methoden schreiben, mit denen Sie von außen die Werte für `friction`, `bounce` usw. setzen können. Überlegen Sie auch, in welcher Art von Applikation Sie diese Klasse einsetzen könnten. Sie könnten zum Beispiel ein Basketballspiel programmieren – was nahe liegt. Oder stellen Sie sich eine Steuerung per Tastatur vor. Sie könnten auch die Ball-Grafik ganz einfach durch eine Automobil-Grafik austauschen und hätten ein Auto-Stuntspiel.

Zeile 4 weisen Sie der `filters`-Eigenschaft des Ball-Movieclips den Schatten zu.

Schritt 23

Film testen

Testen Sie jetzt wieder Ihren Flash-Film. Wie Sie sehen, erzeugt der Ball nun einen Schatten auf dem Boden.

Schritt 24

Noch mehr Bälle



▲ **Abbildung 7**

Die Stärke des Schattens ändert sich in Abhängigkeit von der Entfernung des Balls zum Boden.

Da Sie die gesamte Funktionalität des Balls in einer Klasse realisiert haben, können Sie nun mehrere Bälle erzeugen, die alle dieselben Eigenschaften aufweisen. Ändern Sie dazu das Skript auf der Hauptzeitleiste folgendermaßen:

```
1: for(var i:Number = 0; i<10; i++){
2:     var ball:MovieClip = attachMovie("ball", "ball"+i, i);
3:     ball._x = Stage.width/2;
4:     ball._y = Stage.height/2;
5: }
```

Sie erzeugen nun also in einer `for`-Schleife zehn Bälle. Sie sollten jedoch, bevor Sie den Film testen, die `quality`-Eigenschaft des Schattens in der `BouncingBall`-Klasse auf 1 stellen (siehe Schritt 22). Die Berechnung des Schattens zur Laufzeit ist sehr rechenaufwendig und würde sonst den Film verlangsamen.

Schritt 25

Ergebnis testen

Testen Sie jetzt wieder Ihren Flash-Film. Wie Sie sehen, wimmelt es geradezu von Bällen, die wie wild durch die Gegend hüpfen. Sie können jeden einzelnen Ball anfassen und werfen.

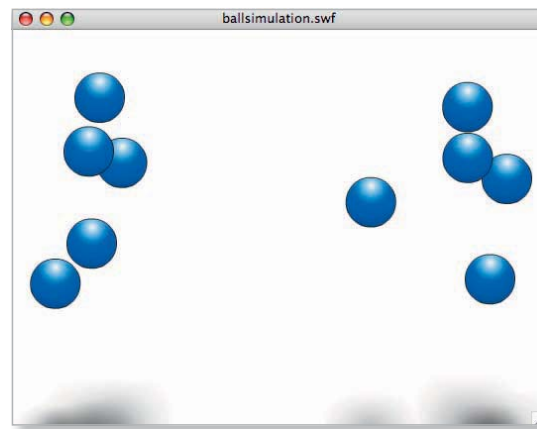


Abbildung 8 ►

Die Bälle werden dynamisch auf der Bühne platziert.

Kabelschwingung

Ein bewegliches Kabel programmieren, das der Mausbewegung folgt

In diesem Workshop werden Sie lernen, wie Sie die Zeichnen-Methoden von Flash einsetzen können, um dynamisch Objekte auf der Bühne zu zeichnen. Sie werden ein Kabel programmieren, das sich nach den Regeln der Physik verhält, wenn Sie es mit der Maus über den Bildschirm schwingen.

Zielsetzungen:

Kabel programmieren

Physikalische Eigenschaften berechnen

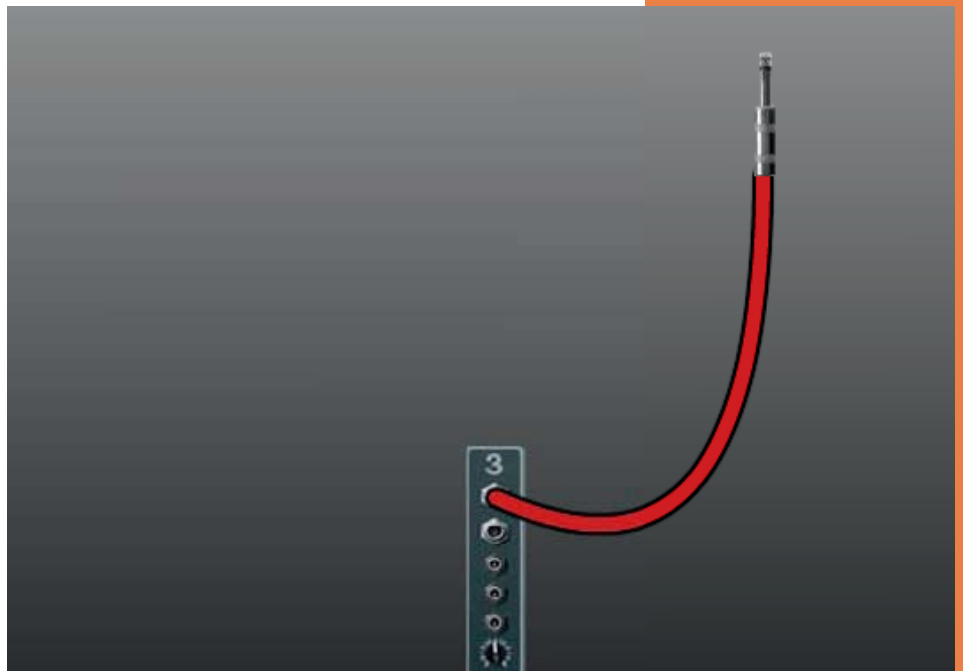
Bild für den Stecker importieren

Stecker am Kabel befestigen

Bewegungen von Stecker und Kabel abstimmen

[Ergebnis: 17_KABELSCHWINGUNG • KABELSIMULATION.SWF]

```
import flash.geom.Point;
var cable:MovieClip = this.createEmptyMovieClip("cable",1);
cable._x = Stage.width/2;
cable._y = Stage.height-100;
var controlPoint:Point = new Point(0,120);
var velocityX:Number = 0;
var velocityY:Number = 0;
var spring:Number = .06;
var friction:Number = .9;
cable.onEnterFrame = function():Void
{
    velocityX += (this._xmouse - controlPoint.x) * spring;
    velocityY += (this._ymouse - controlPoint.y) * spring;
    controlPoint.x += (velocityX * friction);
    controlPoint.y += (velocityY * friction)+120;
    this.clear();
}
```



Schritt 1 Vorbereitungen

Im ersten Schritt legen Sie einen neuen Ordner auf Ihrer Festplatte an und nennen diesen »Kabelsimulation«. Hier werden Sie die von Ihnen erstellten Beispiele zu diesem Workshop abspeichern.

Schritt 2 Dokument erstellen

Erstellen Sie ein neues AS2-Flash-Dokument, und rufen Sie die Dokumenteigenschaften über `[Strg]/[⌘]+[J]` auf. Das Dokument soll die GRÖSSE 550x400 Pixel **1** und eine BILDRATE von 40 bps **3** haben. Die hohe Framerate führt später zu einer flüssigeren Animation, da das Kabel in jedem Frame neu gezeichnet wird. Stellen Sie die HINTERGRUNDFARBE auf #333333 **2**, damit sich das Kabel besser vom Hintergrund abheben wird.

Speichern Sie das Flash-Dokument unter dem Namen KABELSIMULATION.FLA im Projektordner.

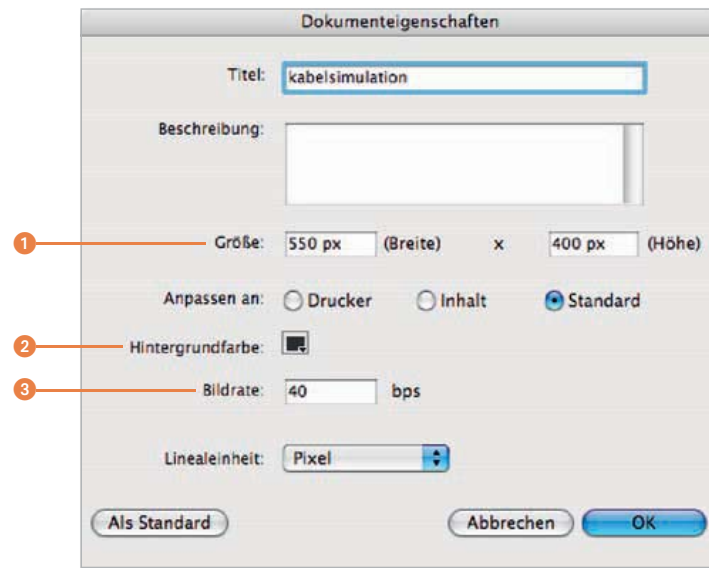


Abbildung 1 ►

Nehmen Sie die im Screenshot gezeigten Einstellungen in den Dokumenteigenschaften vor.

Schritt 3 Movieclip für das Kabel erzeugen

Um eine Kurve mit ActionScript zu zeichnen, benötigen Sie einen Movieclip, in dem Sie zeichnen. Die MovieClip-Klasse besitzt Zeichnen-Methoden, mit denen man geometrische Formen zeichnen kann. Um ein Kabel zu erzeugen, bietet sich die Methode `curveTo()` an. Diese Methode zeichnet, wie der Name schon sagt, eine Kurve.

Klicken Sie in der Zeitleiste auf das erste Bild der ersten Ebene, und öffnen Sie mit `[F9]` bzw. `[Alt]+[F9]` das ActionScript-Fenster. Nun deklarieren Sie zunächst eine Variable `cable` vom Typ `MovieClip`, in der Sie einen mit `createEmptyMovieClip()` erzeugten Movieclip abspeichern:

```
var cable:MovieClip = this.createEmptyMovieClip("cable",1);
cable._x = Stage.width/2;
cable._y = Stage.height-100;
```

Nachdem Sie `cable` erzeugt haben, positionieren Sie diesen Movieclip horizontal zentriert und 100 Pixel oberhalb des unteren Bühnenrandes.

Um eine Kurve zeichnen zu können, müssen Sie zuerst die Linieneigenschaften festlegen. Dazu dient die Methode `lineStyle()`. Übernehmen Sie folgende Codezeile an das Ende Ihres Skripts:

```
cable.lineStyle(8,0xAD0202,100);
```

`lineStyle()` erwartet folgende drei Parameter:

- ▶ Linienstärke in Pixel
- ▶ Linienfarbe als Hexadezimalwert
- ▶ Alphawert

In diesem Fall weisen Sie der Linie eine Stärke von 8 Pixel, eine dunkelrote Farbe und den Alphawert 100 Prozent zu.

Um mit der `curveTo`-Methode eine Kurve zu zeichnen, müssen Sie dieser als Nächstes zwei Ankerpunkte und zwei Steuerpunkte übergeben. Die Ankerpunkte sind die x- und y-Zielkoordinaten, in denen die Kurvenlinie endet. Die Steuerpunkte kann man als Magnete betrachten, die die Krümmung der Linie bestimmen. Der Anfangspunkt entspricht automatisch dem Nullpunkt, es sei denn, Sie haben diesen mit `moveTo()` woanders angesetzt.

Zum Festlegen der Anker- und Steuerpunkte bietet sich die `Point`-Klasse an. Importieren Sie diese in der ersten Zeile Ihres Skripts folgendermaßen:

```
import flash.geom.Point;
```

Legen Sie nun die Anker- und Steuerpunkte fest, und zeichnen Sie mit diesen eine Kurve. Ergänzen Sie dazu ganz unten folgende Codezeilen:

```
var controlPoint:Point = new Point(0,120);
```

Schritt 4

Linieneneigenschaften festlegen

Schritt 5

Kurve zeichnen

Flash-Hilfe

In der Flash-Hilfe finden Sie weitere Erklärungen zu den Zeichnen-Methoden.

```
var anchorPoint:Point = new Point(200,-200);
cable.curveTo(controlPoint.x,controlPoint.y,anchorPoint.x,
    anchorPoint.y);
```

Schritt 6 Film testen

Testen Sie Ihren Flash-Film mit `Strg`/`⌘` + `↵`. Wie Sie sehen, haben Sie entsprechend der angegebenen Punkte eine Kurve gezeichnet. In den nächsten Schritten werden Sie aus dieser Kurve Ihr Kabel erschaffen.



Abbildung 2 ►
Allein mit ActionScript wird eine Kurve gezeichnet.

Schritt 7 Kurve mit der Maus bewegen

Ziel ist es, das Kabel mit der Maus zu bewegen. Der Endpunkt des Kabels soll dabei der Mausposition entsprechen. Zu diesem Zweck verwenden Sie die `onEnterFrame`-Funktion von `cable`, in der Sie den Endpunkt des Kabels in jedem Bild pro Sekunde der Mausposition zuordnen. Programmieren Sie die `onEnterFrame`-Funktion des Kabels folgendermaßen am Ende Ihres Skripts:

```
1: cable.onEnterFrame = function():Void
2: {
3:     this.clear();
4:     this.lineStyle(8,0xAD0202,100);
5:     this.curveTo(controlPoint.x,controlPoint.y,this.
        _xmouse,this._ymouse);
6: }
```

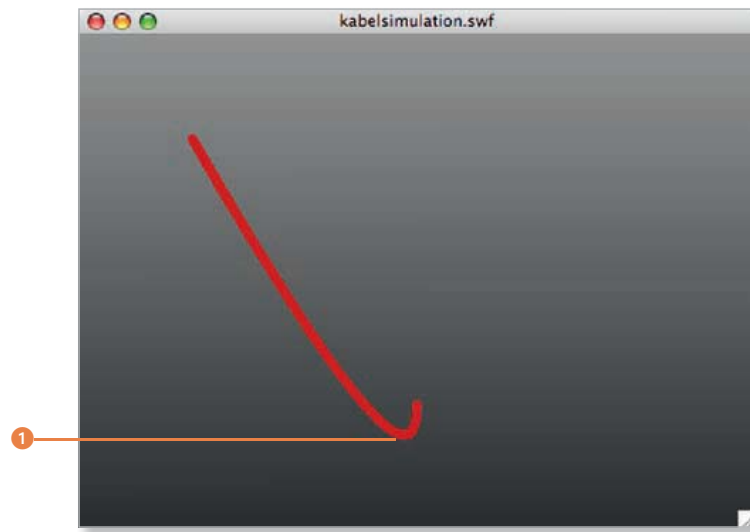
Da nun in jedem Bild eine Kurve gezeichnet wird, müssen Sie die Kurve aus dem Bild zuvor wieder löschen. Andernfalls würden alle diese Kurven sichtbar bleiben – und das wollen Sie ja nicht. Das Löschen erledigen Sie mit dem Aufruf von `clear()` in Zeile 3. Da durch das Löschen jedoch

die Angaben zur Kurvenlinie fehlen, müssen Sie die Linieneigenschaften in Zeile 4 neu definieren.

In Zeile 5 übergeben Sie schließlich der `curveTo`-Methode als Ankerpunkte die `_xmouse`- und `_ymouse`- Eigenschaft von `cabl`.

Testen Sie nun wieder Ihren Flash-Film mit `Strg/⌘+↵`. Wie Sie sehen, wird das Kabel ausgehend von der x- und y-Position von `cabl` zu der Mausposition gezeichnet. Das Kabel scheint jetzt schon eine gewisse Masse zu besitzen, da der y-Steuerpunkt 120 Pixel ❶ unterhalb von `cabl` liegt. Das Kabel scheint damit herunterzuhängen.

Allerdings ist dieses Ergebnis noch nicht wirklich physikalisch korrekt. In den nächsten Schritten werden Sie ein bisschen Bewegung in die Sache bringen.



Schritt 8

Film testen

◀ **Abbildung 3**
Das Kabel bewegt sich mit der Maus, schwingt aber noch nicht nach.

Da durch die Position der Steuerpunkte eine Masse angedeutet wird, werden Sie nun bei der Bewegung des Kabels diese Masse nach physikalischen Regeln bewegen. Wenn man ein flexibles Objekt wie ein Kabel, das an einem Ende befestigt ist, an dem anderen Ende festhält und schnell von einer zur anderen Seite schwingt, wird es nachschwingen bzw. nachfedern. Für diese Bewegung existiert eine Formel: Sie definiert die Beschleunigung proportional zu der Distanz des Zielpunktes, auch Spring genannt:

```
Geschwindigkeit += (Ziel - Position) * Federung;  
Position += (Geschwindigkeit * Reibung);
```

Schritt 9

Physikalische Eigenschaften hinzufügen

Deklariieren Sie *vor* der `onEnterFrame`-Funktion folgende Variablen, um die Formel anwenden zu können:

```
var velocityX:Number = 0;  
var velocityY:Number = 0;  
var spring:Number = .06;  
var friction:Number = .9;
```

In den ersten beiden Zeilen deklarieren Sie die Variablen `velocityX` und `velocityY` für die x- und y-Geschwindigkeit. Die Variable `spring` bestimmt den Federungsfaktor. `friction` ist der Wert der Reibung, durch die die Bewegung nach einer Zeit zum Stehen kommt. Erweitern Sie nun die `onEnterFrame`-Funktion am Anfang mit folgenden Zeilen:

```
1: velocityX += (this._xmouse - controlPoint.x) * spring;  
2: velocityY += (this._ymouse - controlPoint.y) * spring;  
3: controlPoint.x += (velocityX *= friction);  
4: controlPoint.y += (velocityY *= friction)+120;
```

In den Zeilen 1 bis 4 wenden Sie die oben beschriebene Formel an. In Zeile 4 addieren Sie noch den Wert 120 auf den `controlPoint.y`, um weiterhin die Krümmung des Kabels nach unten zu erzeugen.

Schritt 10

Film testen

Testen Sie nun den Flash-Film. Bewegen Sie die Maus hin und her. Die Formel funktioniert! Das Kabel schwingt Ihrer Bewegung hinterher.

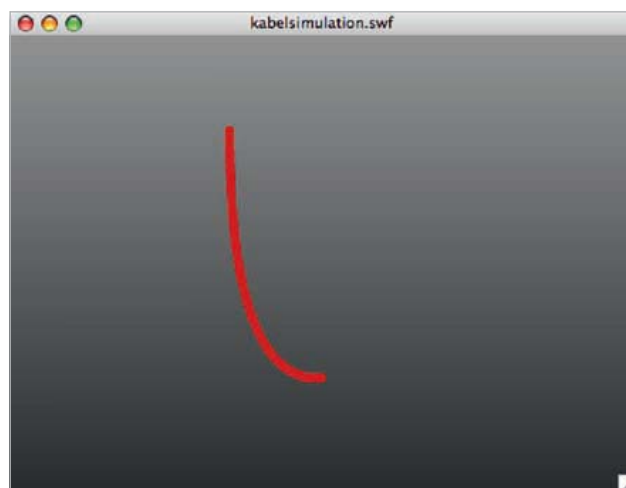


Abbildung 4 ►
Das Kabel schwingt.

Um das Kabel grafisch noch etwas zu verschönern, werden Sie nun eine Außenlinie um das Kabel zeichnen. Da es sich bei der Kurve bereits um eine Linie handelt, steht Ihnen mit Flash nicht die Möglichkeit zur Verfügung, eine weitere Außenlinie um die Linie zu definieren. Dies ist nur Formen wie zum Beispiel einem Kreis oder einem Rechteck vorbehalten. Mit den Möglichkeiten der GlowFilter-Klasse können Sie allerdings diesen Effekt trotzdem erzielen. Erweitern Sie Ihr Skript unter dem bisherigen Code um folgende Zeilen:

```
this.cable.filters = [new flash.filters.GlowFilter(0x000000, 1,  
2, 2, 10, 3, false, false)];
```

Mit dieser Zeile weisen Sie der `filters`-Eigenschaft von `cable` einen `GlowFilter` zu, der einen schwarzen Schein um das Kabel legt. Durch die übergebenen Werte wird dieser Schein wie eine schwarze Linie dargestellt.

Testen Sie erneut Ihren Flash-Film. Sehen Sie: Sie haben die Möglichkeiten von Flash überlistet und eine Kurvenform mit Außenlinien programmiert.



Importieren Sie über DATEI • IMPORTIEREN • IN BÜHNE IMPORTIEREN die Datei `PLUG.PNG`, die Sie in dem Workshop-Ordner auf der beiliegenden Buch-DVD finden.

Klicken Sie auf das Bild, und konvertieren Sie es mit `[F8]` in ein Symbol bzw. in einen Movieclip. Setzen Sie den Registrierungspunkt **2** nach unten in die Mitte. Geben Sie dem Movieclip den NAMEN »plug« **1**, und bestätigen Sie die Eingabe.

Schritt 11

Außenlinie zeichnen

Schritt 12

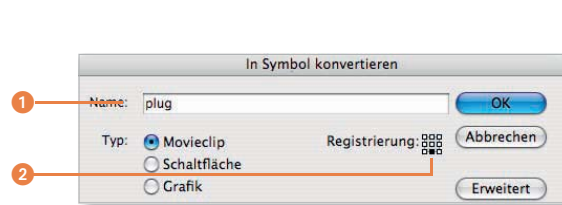
Film testen

◀ Abbildung 5

Die Kurve wirkt durch den GlowFilter plastischer.

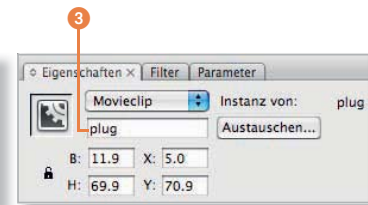
Schritt 13

Einbauen eines Kabelsteckers



▲ **Abbildung 6**

Ein neuer Movieclip wird angelegt.



▲ **Abbildung 7**

Instanzname zuweisen

Markieren Sie nun den neuen Movieclip auf der Bühne, und weisen Sie ihm im Eigenschaftenfenster den Instanznamen »plug« **3** zu.

Schritt 14 Stecker am Kabel befestigen

Ziel ist es nun, den Stecker an dem Kabel zu befestigen und ihn mit dem Kabel mitschwingen zu lassen. Der Stecker gehört an das Ende des Kabels, das der `_xmouse`- und `_ymouse`-Position entspricht. Dies stellt also kein Problem dar.

Um den Schwung des Kabels aufzunehmen, verwenden Sie einen einfachen Trick: Sie errechnen den Winkel zwischen `controlPoint.x` und `_xmouse`. Durch die Federung bewegt sich `controlPoint.x` um die x-Position der Maus hin und her, bis er auf dieser zum Stehen kommt. Während der Bewegung entsteht ein bestimmtes Winkelverhältnis zwischen diesen beiden Punkten. Den Winkel wenden Sie ganz einfach auf die `_rotation`-Eigenschaft des Steckers an, und schon schwingt der Stecker mit.

Das Ganze realisieren Sie in der `onEnterFrame`-Funktion des Steckers `plug`. Übernehmen Sie folgende Zeilen unter dem bisherigen Code in Ihr Skript:

```
1: plug.swapDepths(cable);
2: plug.onEnterFrame = function():Void
3: {
4:     this._x = _xmouse;
5:     this._y = _ymouse;
6:     var targetX:Number = controlPoint.x+cable._x;
7:     var targetY:Number = controlPoint.y+cable._y;
8:     var rotation:Number = Math.atan2(targetY-this._y,
        targetX-this._x);
9:     rotation*=180/Math.PI ;
10:    this._rotation = rotation-90;
11: }
```

In Zeile 1 wechseln Sie zuerst die Ebenentiefe von `plug` und `cable`, damit sich der Stecker über dem Kabel befindet und somit nicht von diesem überdeckt ist.

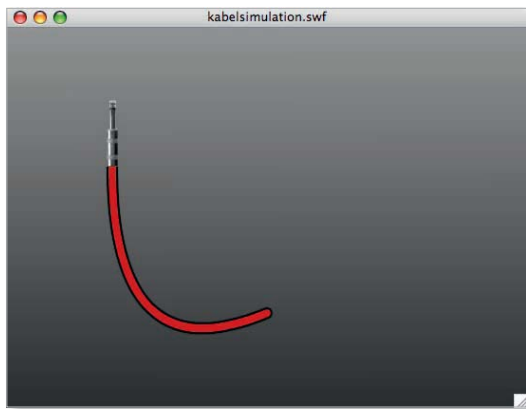
Ebenentiefe (DEPTH) eines Movieclips

Die Ebenentiefe eines Movieclips bestimmt die Überlagerungsreihenfolge aller Movieclips auf der Bühne. Der Movieclip mit der höchsten Tiefe liegt vor allen anderen. Die Tiefe eines Movieclips können Sie mit `Instanzname.getDepth()` abfragen und über `Instanzname.swapDepth(value)` setzen.

Ab Zeile 2 startet die `onEnterFrame`-Funktion von `plug`. In dieser setzen Sie zuerst die Position des Steckers auf die der Mausposition. In den Zeilen 6 und 7 errechnen Sie die x- und y-Position des Steuerpunktes. Diesen müssen Sie neu berechnen, da er relativ zur Mausposition von `cable` bestimmt wird. Um die absolute Position zu erhalten, müssen Sie nur die Position von `cable` zu den Steuerpunkten hinzu addieren.

In der nächsten Zeile errechnen Sie mit Hilfe der `Math.atan2`-Methode das Winkelverhältnis zwischen den Steuerpunkten und der Position des Steckers. Bei dem Rückgabewert von `Math.atan2()` handelt es sich um den Winkel als Radiant. Der `_rotation`-Eigenschaft müssen Sie allerdings den Winkel in Grad zuweisen. In Zeile 9 wandeln Sie deswegen das Bogenmaß in Grad um. Danach, in Zeile 10, weisen Sie der `_rotation`-Eigenschaft von `plug` ihren neuen Wert zu.

Testen Sie nun Ihren Film. Wie Sie sehen, bewegt sich der Kabelstecker entsprechend der Bewegung des Kabels. Sie haben es geschafft! Egal, wie sehr Sie das Kabel schwingen, alles verhält sich nach den von Ihnen definierten physikalischen Regeln.



Da mancher Code nur zur Veranschaulichung der Zeichnen-Methoden gedacht war und das Ergebnis auch nicht mehr beeinflusst, können Sie die Zeilen 5, 7 und 8 des Quelltextes löschen.

```
5: cable.lineStyle(8,0xAD0202,100);
7: var anchorPoint:Point = new Point(200,-200);
8: cable.curveTo(controlPoint.x,controlPoint.y,anchorPoint.x,
    anchorPoint.y);
```

Anwendungsgebiete

Stellen Sie sich vor, wofür Sie diese Applikation einsetzen können. Sie könnten einen MP3-Player programmieren, bei dem zwischen den Liedern gewechselt wird, indem das Kabel mit verschiedenen Steckern verbunden werden muss. Sie können das Programm auch so erweitern, dass Sie eine Seitennavigation für eine Website daraus bauen.

Schritt 15

Film erneut testen

◀ Abbildung 8

Der Stecker am Ende des Kabels schwingt physikalisch korrekt mit.

Schritt 16

Quelltext aufräumen



Auf der Buch-DVD finden Sie den Workshop auch mit ActionScript 3 realisiert.

3D-Programmierung in Flash CS3

3D und Flash – geht das denn?

Um Objekte im dreidimensionalen Raum darzustellen, benötigt man etwas, worüber Flash leider nicht von sich aus verfügt: ein Koordinatensystem mit drei Achsen. Die dritte Achse, **z-Achse** genannt, würde als sogenannte Tiefenachse fungieren. Flash verfügt über zwei Achsen, x horizontal und y vertikal, die ein rechtwinkliges Kreuz bilden. Die z-Achse würde zu diesen im 45-Grad-Winkel von links oben nach rechts unten verlaufen und somit eine Perspektive erzeugen können. Nur woher nehmen wir nun die dritte Achse?

Wie gerne würde ich Ihr Gesicht sehen, wenn Sie im Verlauf dieses Exkurses feststellen werden, wie ausgesprochen simpel es ist, diese fehlende Achse mit ein paar Tricks zu programmieren.

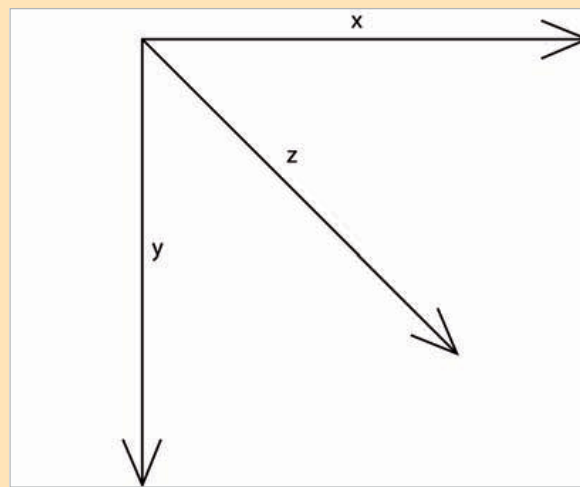


Abbildung 1 ►

Die z-Achse gibt die Tiefe eines Objekts im 3D-Raum an.

3D und 2D – was ist der Unterschied?

Sie haben im Laufe des Buches bereits diverse Anwendungen programmiert, in denen Sie die x- und y-Werte und auch die Skalierungswerte verändert haben, um visuelle Effekte zu erzielen. Was ist nun aber nötig, um dreidimensionale Effekte herzustellen? Die Antwort: Man muss eine künstliche Perspektive erzeugen.

Das Geschehen im dreidimensionalen Raum übersetzt »in Flash« lässt sich leicht erklären: Wenn sich ein Objekt von Ihnen wegbewegt, wird es

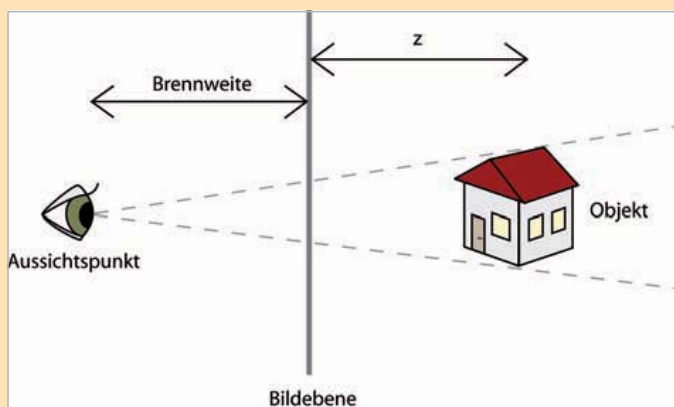
kleiner. Umgekehrt wird es größer. Das heißt, man muss die **Skalierung** anpassen. Hinzu kommt, dass sich das Objekt auf den perspektivischen **Fluchtpunkt** zubewegt. Man muss also auch den x- und den y-Wert verändern. Ein Objekt in weiter Ferne ist auch nicht so gut zu erkennen wie eines, das sich genau vor Ihnen befindet. Nicht nur, weil das entfernte Objekt kleiner erscheint, sondern weil es weiter vorne auch heller ist und damit sattere Farben hat. Man könnte in Flash den **Alpha-Wert** anpassen, um dies zu erreichen.

Zusätzlich muss eines beachtet werden: Auch wenn Sie die Skalierungswerte, die x- und y-Position und Alpha geschickt anpassen, so kann man in Flash lediglich die Illusion eines dreidimensionalen Raums erschaffen – die Objekte liegen immer noch flach auf einer Ebene. Um nun aber anzuzeigen, dass sich das kleiner skalierte Objekt weiter im Hintergrund befindet als ein anderes, muss zusätzlich die **Tiefeneigenschaft** des Movieclips angepasst werden. Sonst würden die Objekte sich falsch überlagern und die Illusion damit sterben.

Wenn man noch genauer vorgehen wollte, müsste sich das Objekt auch noch zum Fluchtpunkt krümmen. Uns reicht es aber vorerst, die Grundlagen zu verstehen.

Die Berechnung

Zur Berechnung der Perspektive benötigt man drei Faktoren: zunächst den Aussichtspunkt, von dem aus man ein Objekt betrachtet. Zwischen diesem Punkt und dem Objekt liegt die Bildebene. Unter dieser kann man sich ganz einfach den Computerbildschirm vorstellen. Der Bereich zwischen Aussichtspunkt und Bildebene ist die Brennweite. Zwischen der Bildebene und dem Objekt liegt der z-Wert. Je größer er ist, desto weiter soll sich das Objekt entfernen, also kleiner werden.



◀ **Abbildung 2**
Grundlagen zur Berechnung
der Perspektive

Diese Werte in einer Formel vereint genügen, um die perspektivische Berechnung durchzuführen:

```
scale = focalLength / (focalLength + z)
```

`focalLength` ist dabei die Brennweite. Nehmen wir an, wir geben der Variablen `focalLength` einen Wert von 250. Wenn jetzt der z-Wert 0 ist, ergibt sich folgende Gleichung:

```
scale = 250 / (250+0) // = 1
```

Wenn `z` den Wert 0 hat und wir also erwarten, dass sich das Objekt direkt vor uns befindet, ergibt die Formel einen Wert von 1. Setzen wir nun das Objekt etwas nach hinten, auf den z-Wert 250, erhalten wir folgenden Wert:

```
scale = 250 / (250+250); // = 0.5
```

Das Objekt ist nur noch halb so groß. Wie Sie sehen, ist der Wert der Brennweite entscheidend für die Positionierung und Darstellung im dreidimensionalen Raum.

Z-Sorting

Wie bereits erwähnt, müssen die Objekte auf der Bühne korrekt in der Tiefe verteilt werden, da sie sich sonst nicht entsprechend ihrer räumlichen Tiefe überlagern. Und mit diesem Satz ist im Grunde genommen schon die Antwort auf die Frage, wie man das anstellen soll, gegeben: Nichts liegt näher, als den z-Wert des Objekts als Tiefe zu verwenden. Je größer der z-Wert ist, desto weiter hinten befindet sich das Objekt. Da das Tiefenmanagement in Flash andersherum funktioniert, also das Objekt mit der höchsten Tiefe sich ganz oben befindet, nimmt man einfach den negativen z-Wert als Tiefe.

In **ActionScript 2** wäre dies ganz einfach mit `swapDepth(-z)` erledigt. In ActionScript 2 kann man einem Movieclip in einem gewissen Rahmen eine beliebige Tiefe zuweisen.

In **ActionScript 3** hat sich dies allerdings grundlegend geändert. Die Objekte, die auf der Bühne erzeugt werden, können nur die nächsthöhere oder eine bereits belegte Tiefe annehmen. Wenn Sie also per `addChild()` zwei Objekte nacheinander auf der Bühne instantiieren, werden diese auf den Ebenen 0 und 1 angelegt. Sie könnten jetzt kein weiteres Objekt auf Ebene 100 erzeugen, sondern wie gesagt nur auf einer bereits belegten Ebene oder auf der nächsthöheren. Auch wenn Sie drei Objekte auf der

Bühne haben und dasjenige mit der mittleren Tiefe löschen, werden die Tiefenwerte automatisch neu generiert. Das Objekt mit der Tiefe 2 würde nun automatisch die Tiefe 1 bekommen.

`DisplayObject` stellt drei Methoden zur Bestimmung der Tiefe des Objekts zur Verfügung:

- ▶ `setChildIndex(child:DisplayObject, index:int)`
Diese Methode setzt den Index eines ChildObjects auf eine Tiefe im möglichen Bereich.
- ▶ `swapChildren(child1:DisplayObject, child2:DisplayObject)`
Diese Methode tauscht die Tiefen zweier Objekte aus. Lag das eine Objekt vorher vor dem anderen, ist dies nach Anwenden der Methode genau andersherum.
- ▶ `swapChildrenAt(index1:int, index2:int)`
Diese Methode vertauscht die Tiefe zweier Objekte unter Angabe der Tiefenwerte.

Wie Sie sehen, müssen wir uns etwas einfallen lassen, um denselben Effekt zu erreichen, den wir mit ActionScript 2 mit einer Zeile Code erreicht haben. Eine einfache Lösung für dieses Problem ist, alle Objekte in einem Array zu speichern, dieses nach dem z-Wert der Objekte zu sortieren und dementsprechend die Tiefe mit der `setChildIndex`-Methode anzugleichen.

```
1: array.sortOn("z", Array.DESENDING | Array.NUMERIC);
2: for(var i:uint = 0; i < array.length; i++) {
3:   var object3D:MovieClip = array[i];
4:   setChildIndex(object3D, i);
5: }
```

In Zeile 1 sortieren Sie das Array nach der z-Eigenschaft der darin gespeicherten Objekte in absteigender Reihenfolge. Das Objekt mit dem höchsten z-Wert befindet sich also am Anfang des Arrays. Nun durchlaufen Sie das Array in einer `for`-Schleife und weisen den Objekten als Tiefenwert den Wert `i` zu. Da dieser bei 0 anfängt, weisen Sie also dem Objekt mit dem höchsten z-Wert die geringste Tiefe zu. Damit liegen die Objekte entsprechend ihrer Tiefe z übereinander angeordnet. Und das war es auch schon! Nun sind Sie genug vorbereitet, um Ihre erste 3D-Anwendung zu programmieren.

3D-Sternenfeuer

Eine 3D-Engine programmieren

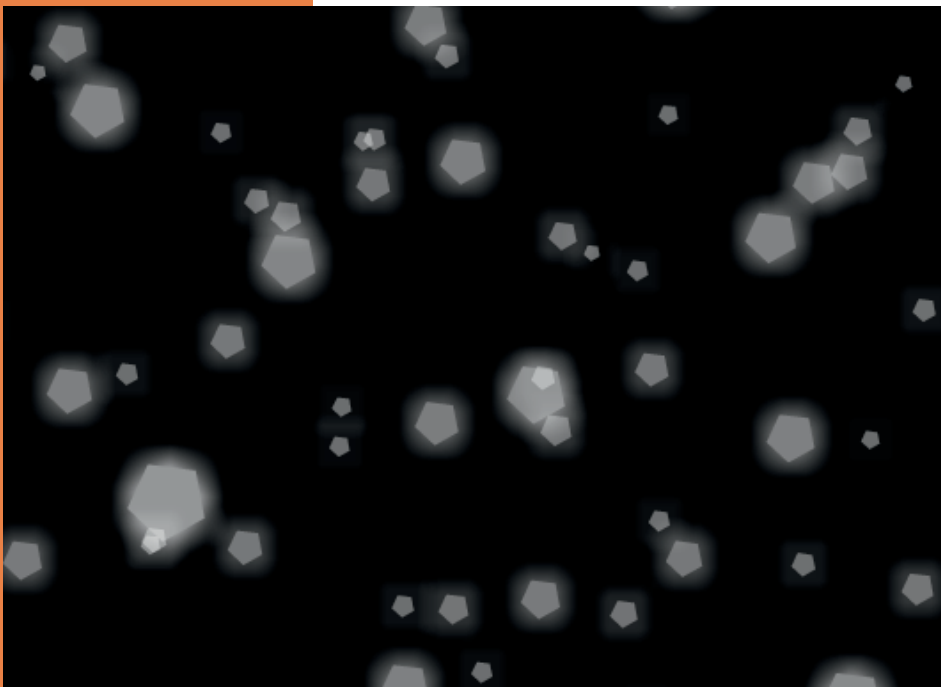
```
package{
    import flash.display.Sprite;
    import flash.display.Loader;
    import flash.net.URLRequest;
    import flash.events.*;
    import flash.display.Bitmap;
    public class Picture extends Sprite
    {
        public var posZ:Number;
        public var posX:Number;
        public var posY:Number;
        public var speedZ:Number;
        public var startX:Number;
        public var startY:Number;
        public function Picture(url:String):void
        {
```

Dieser Workshop dient zur Einführung in die 3D-Programmierung in Flash. Sie werden lernen, wie Sie Objekte im Raum erzeugen und bewegen. Sie werden Sterne in den Himmel werfen.

Zielsetzungen:

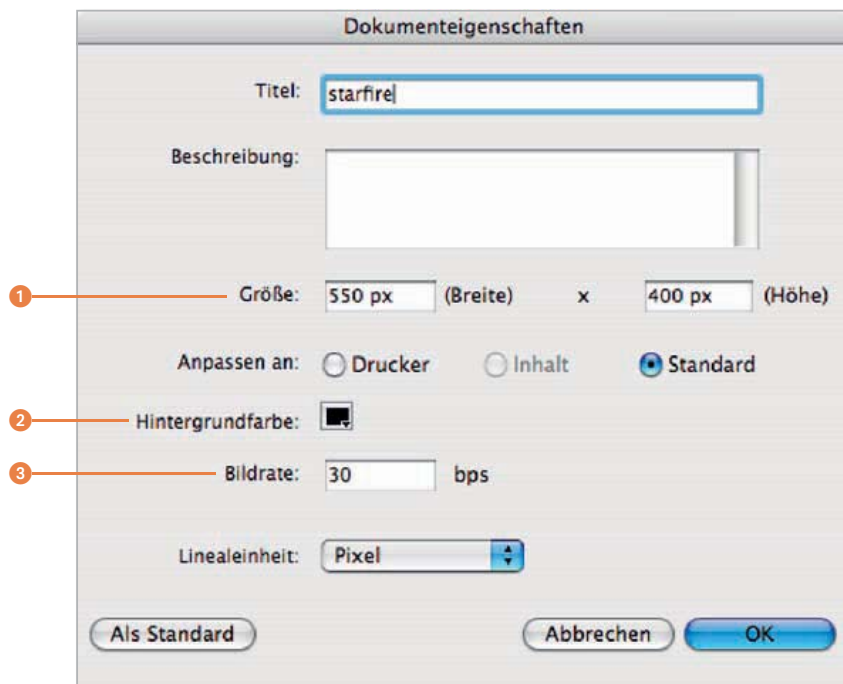
- Stern zeichnen
- 3D-Engine programmieren
- Sterne ins All werfen

[Ergebnis: 18_3D-STERNEN-
FEUER • STARFIRE.SWF]



Im ersten Schritt legen Sie einen neuen Ordner auf Ihrer Festplatte an und nennen ihn STERNENFEUER. Hier werden Sie die von Ihnen erstellten Dateien zu diesem Workshop abspeichern.

Erstellen Sie ein neues AS3-Flash-Dokument mit der Größe 550x400 Pixel ❶ und einer BILDRADE von 30bps ❸. Stellen Sie die HINTERGRUNDFARBE auf Schwarz ❷. Speichern Sie das Dokument unter dem Namen STARFIRE.FLA in dem Projektordner ab.



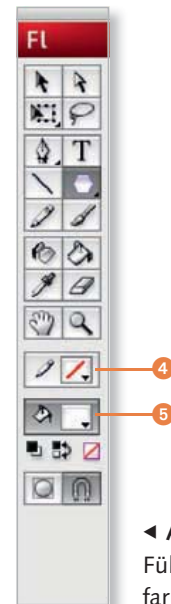
Wählen Sie das Polysternwerkzeug  aus der Werkzeugpalette. Stellen Sie die Füllfarbe ❺ auf Weiß, und deaktivieren Sie die Außenlinien ❹.

Zeichnen Sie nun einen Stern mit einer ungefähren Breite und Höhe von 200 Pixel auf die Bühne.

Markieren Sie den Stern mit der Maus, und konvertieren Sie ihn mit **[F8]** in ein Symbol bzw. in einen Movieclip. Geben Sie diesem den Namen »Star« ❻. Positionieren Sie den Registrierungspunkt mittig ❼. Aktivieren Sie die Checkbox EXPORT FÜR ACTIONSCRIPT ❽. Das Feld KLASSE sollte automatisch den Wert »Star« haben. Bestätigen Sie Ihre Eingaben.

Schritt 1 Vorbereitungen

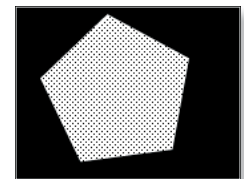
Schritt 2 Dokument erstellen



◀ **Abbildung 1**
Füll- und Strich-
farbe zuweisen

◀ **Abbildung 2**
Dokumenteigenschaften

Schritt 3 Stern-Movieclip erzeugen



▲ **Abbildung 3**
Breite und Höhe des Sterns
sollten bei ca. 200 Pixel liegen.

Fehlermeldung

Da Sie dem Movieclip eine Klasse zugeordnet haben, die nicht vorhanden ist, wird Flash eine Warnung anzeigen. Diese können Sie jedoch einfach ignorieren, da der Bezeichner in diesem Fall nur dazu dient, eine Instanz des Movieclips erzeugen zu können. Die Klasse selbst muss nicht vorhanden sein.

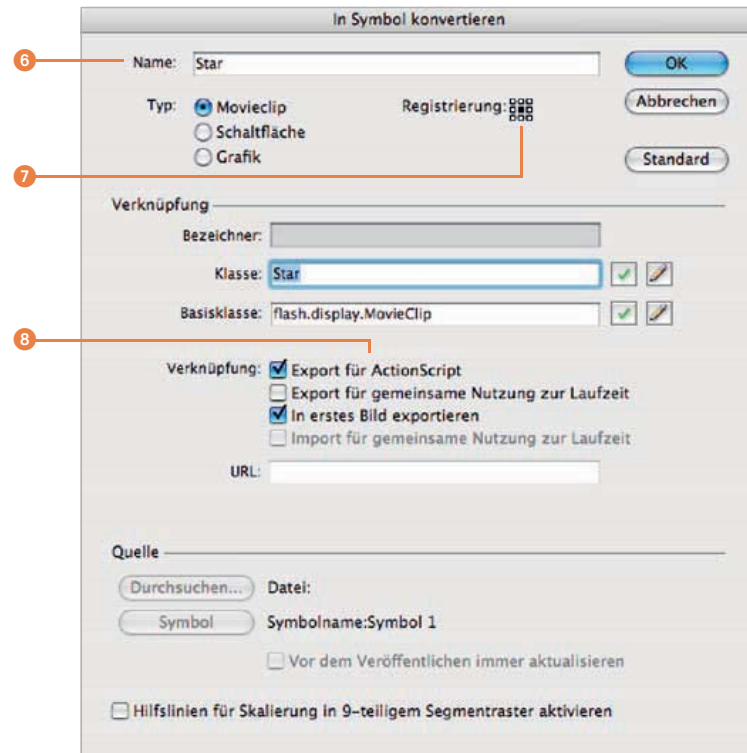


Abbildung 4 ►

In Symbol konvertieren

Der Stern-Movieclip befindet sich nun in Ihrer Bibliothek. Da die Sterne dynamisch erzeugt werden sollen, löschen Sie den Stern-Movieclip wieder von der Bühne.

Schritt 4 Dokumentklasse anlegen

Seit Flash CS3 ist es möglich, jedem ActionScript 3-Flash-Dokument direkt eine ActionScript-Klasse zuzuordnen – die Dokumentklasse. Wenn Sie dann später die Flash-Datei veröffentlichen, wird eine Instanz dieser Klasse als Objekt der obersten Ebene dieser swf-Datei erstellt und der Konstruktor dieser Klasse aufgerufen.

Eine solche Klasse wird in diesem Workshop die allgemeine Steuerung der Flash-Datei übernehmen. Erstellen Sie zu diesem Zweck mit dem Tastaturbefehl **[Strg]/[⌘]+[N]** eine neue ACTIONSCRIPT-DATEI, nennen Sie diese STARFIRE.AS, und speichern Sie die Klasse im Projektordner.

Schritt 5 Dokumentklasse festlegen

Jetzt, da Sie den Klassen-Ordner mit der Flash-Datei verknüpft haben, können Sie im Eigenschaftenfenster, das Sie über **[Strg]/[⌘]+[F3]** aufrufen, die DOKUMENTKLASSE »StarFire« eintragen. Bestätigen Sie die Eingabe mit **[↵]**, damit Flash die Änderungen annimmt.



◀ Abbildung 5

Sie können die Dokumentklasse direkt im Eigenschaftenfenster angeben.

Öffnen Sie die Datei STARFIRE.AS mit Ihrem bevorzugten Editor, und programmieren Sie folgendermaßen den Klassenrumpf:

```
1: package{
2:     import flash.display.MovieClip;
3:     public class StarFire extends MovieClip
4:     {
5:         public function StarFire()
6:         {
7:         }
8:     }
9: }
```

Die Dokumentklasse von `MovieClip` oder `Sprite` erbt in der Regel.

Später ist es unser Ziel, die Sterne per Mausklick zu erzeugen. Dies wird über ein Maus-Event gesteuert. Zusätzlich benötigen Sie ein `ENTER_FRAME`-Event, in dem Sie die Eigenschaften des Sterns in jedem Bild pro Sekunde verändern. Dazu müssen Sie das `flash.events`-Package importieren. Der Stern soll natürlich auch leuchten. Importieren Sie dafür den `GlowFilter` unter der ersten Import-Anweisung:

```
import flash.events.*;
import flash.filters.GlowFilter;
```

Wenn Sie den Exkurs über 3D-Programmierung bereits gelesen haben, wissen Sie, dass Sie für einfache 3D-Programmierung nur wenige Variablen benötigen. Deklarieren Sie nun über dem Konstruktor die `focalLength`-Variable sowie die Aussichtspunkt-Variablen:

```
private var focalLength:int = 250;
private var viewPointX:int = stage.stageWidth/2;
private var viewPointY:int = stage.stageHeight/2;
```

Schritt 6

Die StarFire-Klasse

Schritt 7

Imports vornehmen

Schritt 8

Variablen deklarieren

3D-Programmierung

Eignen Sie sich die Grundlagen der 3D-Programmierung in Flash im Exkurs auf Seite 184 an.

Schritt 9

MouseListener hinzufügen

In der ersten Zeile deklarieren Sie die Variable für die Brennweite: `focalLength` und initialisieren diese mit dem Wert 250. In den nächsten Zeilen legen Sie die Koordinaten des Aussichtspunkts in den Variablen `viewPointX` und `viewPointY` auf die Bühnenmitte.

Da die Sterne per Mausklick auf der Bühne erzeugt werden sollen, müssen Sie der Bühne (`stage`) einen Maus-Listener hinzufügen. Bei einem Klick-Event soll eine Methode `mouseDownListener()` aufgerufen werden. Fügen Sie der Bühne den entsprechenden Listener im Konstruktor der `StarFire`-Klasse hinzu:

```
stage.addEventListener(MouseEvent.CLICK, mouseDownListener);
```

Schritt 10

Sterne erzeugen

Schreiben Sie nun die `mouseDownListener`-Methode, die bei jedem Mausklick aufgerufen wird, folgendermaßen unterhalb des Konstruktors:

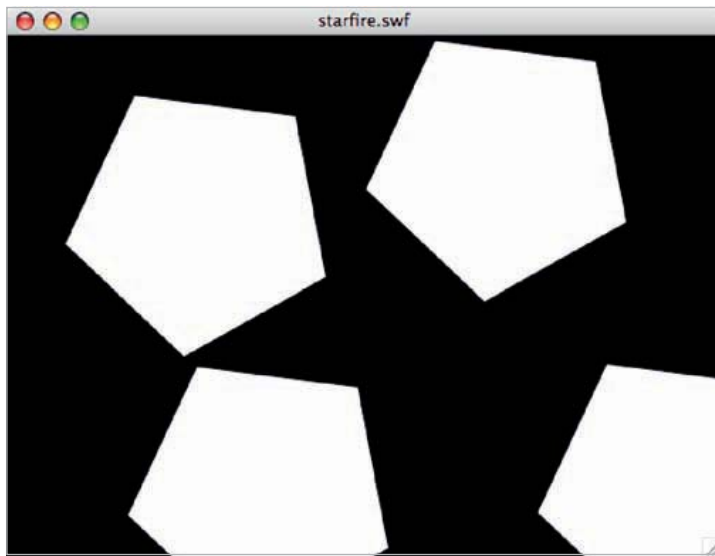
```
1: private function mouseDownListener(event:MouseEvent):void
2: {
3:     var star:Star = new Star();
4:     addChild(star);
5:     star.x = mouseX;
6:     star.y = mouseY;
7: }
```

In Zeile 3 speichern Sie in der Variablen `star` eine neue Instanz des `StarMovieclip` und fügen diesen in Zeile 4 der Bühne hinzu. Mit den nächsten beiden Zeilen positionieren Sie den Stern auf die Mausposition.

Schritt 11

Film testen

Testen Sie nun Ihren Flash-Film mit `Strg`/`⌘`+`↵`. Klicken Sie auf die Bühne. Wie Sie sehen, entsteht immer genau dort, wo Sie klicken, ein neuer Stern. Die Sterne erscheinen jetzt sehr groß. Wenn sie aber später in den 3D-Raum hineingeworfen werden, werden sie sich entsprechend ihrer Entfernung verkleinern.



◀ **Abbildung 6**

Per Mausklick entstehen Sterne.

Um später die im 3D-Exkurs genauer beschriebene Formel zum Errechnen der Position und Skalierung des Sterns anzuwenden, müssen Sie zunächst ein paar Variablen festlegen. Diese Variablen speichern die x-, y- und z-Positionen des Sterns im 3D-Raum. Werden diese Werte in die Formel eingesetzt, erhalten Sie die tatsächlichen x-, y- und Skalierungswerte in dem eigentlich zweidimensionalen Koordinatensystem von Flash. Weisen Sie also dem neu erzeugten Stern (`star`) innerhalb der `mouseDownListener`-Methode folgende Eigenschaften zu:

```
1: star.posZ = 0;
2: star.posX = mouseX - viewPointX;
3: star.posY = mouseY - viewPointY;
4: star.speedZ = 100;
```

Als Erstes legen Sie den Wert für `posZ`, der die Position auf der z-Achse bestimmt, auf 0 fest. Der Stern soll sich direkt vor Ihnen befinden, bevor er seinen Weg in den unendlichen Raum beschreitet. Danach setzen Sie `posX` und `posY`, die x- und y-Positionen im 3D-Raum, auf die Mausposition abzüglich der Aussichtspunkt-Koordinaten. Dies hat zur Folge, dass die Sterne auch nach der Umrechnung der 3D-Position in die tatsächliche 2D-Position immer noch auf der Mausposition sind.

In Zeile 4 legen Sie in einer Variablen `speedZ` den Wert 100 fest. `speedZ` soll die Geschwindigkeit bestimmen, mit der der Stern ins All geschossen wird.

Schritt 12

3D-Eigenschaften festlegen

Es gilt die Regel:

Aus der Differenz zwischen 3D-Position und Aussichtspunkt-Koordinaten ergibt sich die tatsächliche x- und y-Position – allerdings nur unter der Voraussetzung, dass die z-Position 0 beträgt.

Schritt 13

Die 3D-Engine

Die 3D-Eigenschaften des Sterns müssen in jedem Bild pro Sekunde in die tatsächlichen 2D-Eigenschaften umgerechnet werden. Dies erledigen Sie am besten in einer Methode, die in jedem Bild neu aufgerufen wird.

Fügen Sie zu diesem Zweck dem `ENTER_FRAME`-Event jeder `Star`-Instanz einen `EventListener` hinzu, der eine Methode `enterFrameListener()` aufruft. Übernehmen Sie dazu in die letzte Zeile der `mouseDownListener`-Methode folgende Anweisung:

```
star.addEventListener(Event.ENTER_FRAME, enterFrameListener);
```

Programmieren Sie nun die Methode `enterFrameListener()` gleich unter die `mouseDownListener`-Methode:

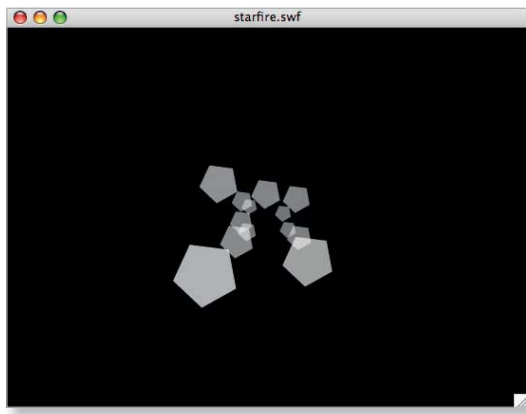
```
1: private function enterFrameListener(event:Event):void
2: {
3:     var star:Star = Star(event.target);
4:     star.posZ += star.speedZ;
5:     var scale:Number = focalLegth/(focalLegth + star.posZ);
6:     star.scaleX = star.scaleY = scale;
7:     star.x = viewPointX + star.posX * scale;
8:     star.y = viewPointY + star.posY * scale;
9:     star.alpha = scale+.6;
10: }
```

In Zeile 3 speichern Sie als Erstes das `target` des Event-Objekts `event` in einer Variablen `star`. Bei dem `target` handelt es sich um die Instanz der `Star`-Klasse, die dieses Event ausgelöst hat. Daraufgehend addieren Sie den `speedZ`-Wert von `star` auf dessen `posZ`-Eigenschaft. Diese wird nun also in jedem Bild um 100 vergrößert, und der Stern fliegt in den Raum. In Zeile 5 wenden Sie die 3D-Formel an und errechnen den `scale`-Wert. Nähere Informationen hierzu können Sie in dem Exkurs zur 3D-Programmierung nachlesen.

Nun setzen Sie die `scaleX`- und `scaleY`-Eigenschaft von `star` auf den `scale`-Wert. In den nächsten zwei Zeilen errechnen Sie aus den 3D-Eigenschaften `posX` und `posY` die tatsächliche Position. Hierzu ist nichts weiter zu tun, als die 3D-Positionswerte mit `scale` zu multiplizieren und zu den entsprechenden Werten des Aussichtspunktes zu addieren. In Zeile 9 setzen Sie auch die `alpha`-Eigenschaft des Sterns auf den `scale`-Wert, damit der Stern weiter im Hintergrund weniger sichtbar sein wird. Da die Sterne allerdings irgendwann nicht mehr zu sehen wären, weil der `scale`-Wert gegen 0 geht, addieren Sie noch 0.6 hinzu. So wird der Stern nicht durchsichtiger als 60%.

Das war es schon! Diese paar Zeilen Berechnung haben es in sich, wie Sie sehen werden, wenn Sie Ihren Film nun testen.

Testen Sie nun Ihren Flash-Film mit `Strg`/`⌘` + `↵`. Klicken Sie auf die Bühne. Wow! Die Sterne fliegen in den endlosen Raum, bis man sie kaum noch sehen kann. Wie Sie feststellen werden – gerade wenn Sie an den äußeren Bühnenrändern klicken – bewegen sich die Sterne direkt auf den Mittelpunkt der Bühne zu, also auf den von Ihnen festgelegten Aussichtspunkt. Im nächsten Schritt werden Sie die Sterne im Raum verteilen.



Schritt 14

Film testen

◀ **Abbildung 7**
Die Sterne fliegen im 3D-Raum.

Damit die Sterne eine andere Richtung einnehmen als die Mitte der Bühne, werden Sie in diesem Schritt einen Geschwindigkeitswert für `posX` und `posY` erzeugen, wie Sie es mit der `z`-Position bereits getan haben.

Fügen Sie zu diesem Zweck in der `mouseDownListener`-Methode unter der Zeile, in der Sie `speedZ` bestimmt haben, folgende Zeilen ein:

```
star.speedX = (mouseX-viewPointX)/2;  
star.speedY = (mouseY-viewPointY)/2;
```

Mit diesen Zeilen erzeugen Sie die Stern-Eigenschaften `speedX` und `speedY`, denen Sie einen Wert zuweisen, der sich aus dem Abstand der Maus zum Aussichtspunkt, also zur Bühnenmitte ergibt. Da dieser Wert etwas zu viel Schwung in die Sache bringen würde, teilen Sie das Ergebnis noch durch 2. Sie erschaffen damit eine Flugbahn, die je nach Klickposition unterschiedlich sein wird.

Was Sie nun noch tun müssen, ist, diese Werte mit in die Berechnung der Position in der `enterFrameListener`-Methode aufzunehmen. Schrei-

Schritt 15

Sterne am Himmel verteilen

ben Sie dort unter die Zeile, in der Sie die z-Position `posZ` berechnen, folgende zwei Codezeilen:

```
star.posX += star.speedX;
star.posY += star.speedY;
```

So addieren Sie den `speedX`- und den `speedY`-Wert auf die 3D-x- und -y-Positionen.

Schritt 16

Film erneut testen

Testen Sie nun Ihren Flash-Film. Klicken Sie mit der Maus in verschiedene Bereiche der Bühne. Wie Sie sehen, schießen die Sterne nun in alle möglichen Richtungen und Sie können diese so über den gesamten Himmel verteilen.

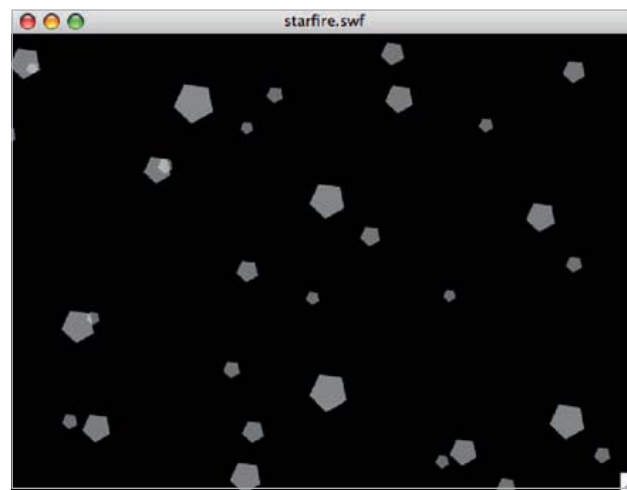


Abbildung 8 ►
Die Sterne werden im
Raum verteilt.

Schritt 17

Sterne vom Himmel holen

Da der `enterFrameListener` nicht wieder gelöscht wird, werden die Position und die Skalierung der Sterne immer weiter berechnet. Da irgendwann die Sterne so klein skaliert sind, dass man sie nicht mehr sehen kann, können Sie Ihrem Computer unnötigen Rechenaufwand ersparen, indem Sie eine z-Position als Grenze einführen, an der der `enterFrameListener` wieder entfernt und der Stern-Movieclip gelöscht wird. Schreiben Sie dazu in die letzte Zeile der Methode `enterFrameListener()`:

```
39: if (star.posZ>10000) {
40:     star.removeEventListener(Event.ENTER_FRAME, enterFrameListener);
41:     removeChild(star);
42: }
```

Sie fragen damit ab, ob die z-Position des Sterns größer als 10000 ist und löschen in diesem Fall den EventListener und den Stern-Movieclip.

Wenn Sie den Film nun testen, würden Sie kaum einen sichtbaren Unterschied feststellen, da die Sterne vorher gegen Null skaliert wurden und so auch irgendwann verschwunden sind. Sie haben jedoch die Performance optimiert.

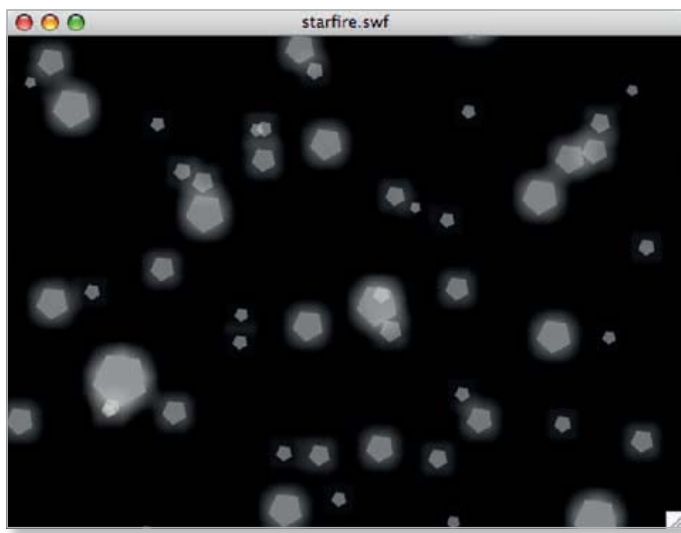
Um die Sterne leuchten zu lassen, fügen Sie nun der Filtereigenschaft eines jeden Sterns (`star`) einen `GlowFilter` hinzu. Schreiben Sie in die `mouse-DownListener`-Methode, bevor Sie `star` den `ENTER_FRAME`-EventListener hinzufügen, folgende Zeile:

```
star.filters = [new GlowFilter(0xffffffff,1,30,30)];
```

Damit weisen Sie dem Stern einen `GlowFilter` zu, der ein weißes Glühen um den Stern erzeugt. Über weitere Möglichkeiten von `GlowFilter` können Sie sich in der Flash-Hilfe informieren.

Testen Sie nun Ihren Flash-Film. Klicken Sie mit der Maus in verschiedene Bereiche der Bühne. Wie Sie sehen, schießen die Sterne leuchtend in alle möglichen Richtungen und verglühen schließlich, wenn sie die 10000er Position auf der z-Achse erreicht haben.

Herzlichen Glückwunsch, Sie haben eine kleine, aber feine 3D-Anwendung programmiert!



Schritt 18

Sternenleuchten

Schritt 19

Film testen

◀ **Abbildung 9**
Unendliche Weiten ...

3D-Bildergalerie

Im Raum fliegende Bilder

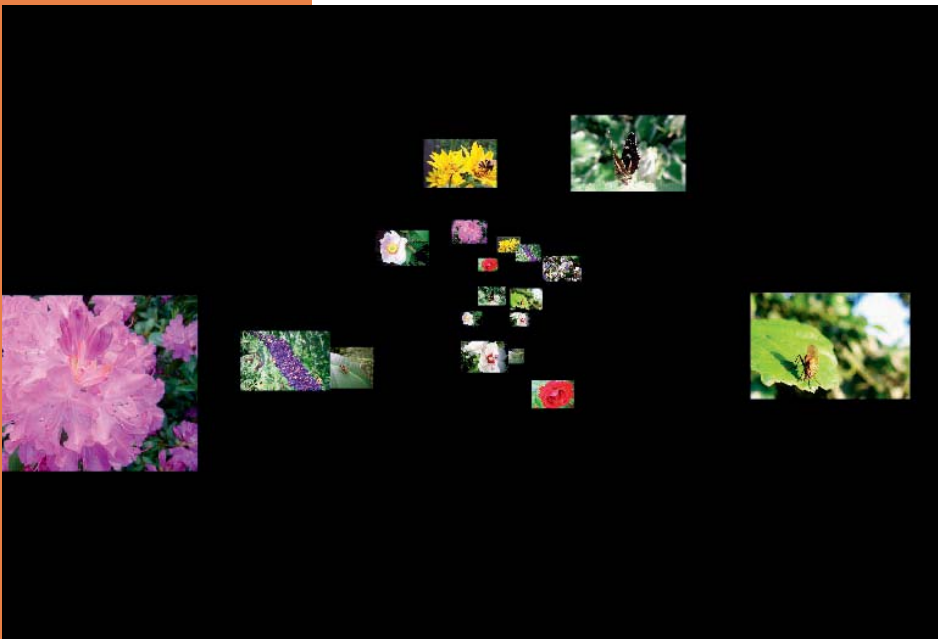
```
package{
    import flash.display.Sprite;
    import flash.display.Loader;
    import flash.net.URLRequest;
    import flash.events.*;
    import flash.display.Bitmap;
    public class Picture extends Sprite
    {
        public var posZ:Number;
        public var posX:Number;
        public var posY:Number;
        public var speedZ:Number;
        public var startX:Number;
        public var startY:Number;
        public function Picture(url:String):void
        {
            var urlRequest:URLRequest = new URLRequest(url);
```

In diesem Workshop werden Sie eine Bildergalerie der besonderen Art programmieren. Jede Wette, dass Sie das Ergebnis mehr als begeistern wird! Ziel ist es, Ihre 3D-Kenntnisse noch etwas zu erweitern und einen dreidimensionalen Raum zu erschaffen, in dem Sie Bilder verteilen und schweben lassen. Sie können in den Raum hineinfahren, um sich die Bilder in voller Größe anzusehen, oder sich einfach langsam treiben lassen.

Zielsetzungen:

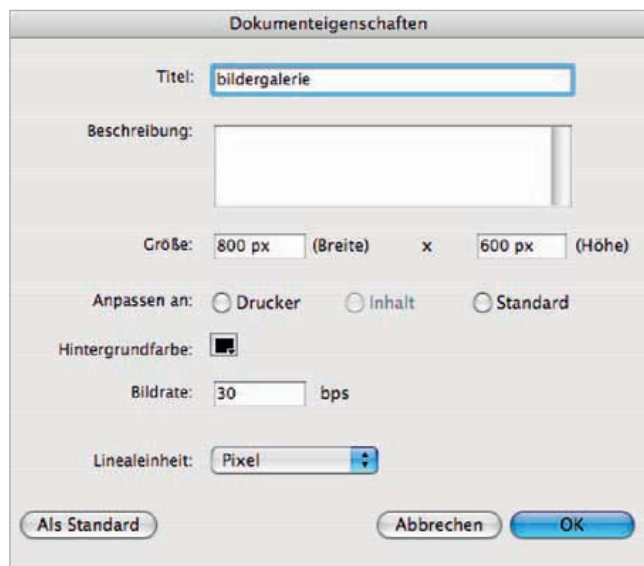
- Bilder laden
- Bilder im dreidimensionalen Raum verteilen
- Bilder schweben lassen
- Kamerafahrt programmieren

[Ergebnis: 19_3D-BILDERGALERIE • BILDERGALERIE.SWF]



Im ersten Schritt legen Sie einen neuen Ordner auf Ihrer Festplatte an und nennen ihn 3D-BILDERGALERIE. Hier werden Sie die von Ihnen erstellten Dateien zu diesem Workshop abspeichern.

Erstellen Sie ein neues AS3-Flash-Dokument mit der Grösse 800x600 Pixel und einer BILDRATE von 30bps. Stellen Sie die HINTERGRUNDFARBE auf Schwarz. Speichern Sie die Datei unter dem Namen BILDERGALERIE.FLA im Projektordner.



Seit Flash CS3 ist es möglich, jedem AS3-Flash-Dokument direkt eine ActionScript-Klasse zuzuordnen – die Dokumentklasse. Wenn Sie dann später die Flash-Datei veröffentlichen, wird eine Instanz dieser Klasse als Objekt der obersten Ebene dieser swf-Datei erstellt und der Konstruktor dieser Klasse aufgerufen. Eine solche Klasse wird in diesem Workshop die allgemeine Steuerung der Flash-Datei übernehmen.

Erstellen Sie mit dem Tastaturbefehl `Strg/⌘+N` eine neue ACTIONSCRIPT-DATEI. Nennen Sie diese PICTUREGALLERY3D.AS, und speichern Sie die Klasse im Projektordner.

Tragen Sie nun im Eigenschaftsfenster, das Sie über `Strg/⌘+F3` aufrufen, als DOKUMENTKLASSE »PictureGallery3D« ein. Bestätigen Sie die Eingabe mit `↵`.

Schritt 1

Vorbereitungen

Schritt 2

Flash-Dokument erstellen

◀ **Abbildung 1**
Die Bildergalerie bekommt einen schwarzen Hintergrund.

Schritt 3

Dokumentklasse anlegen

Schritt 4

Dokumentklasse festlegen

Abbildung 2 ►

Im Eigenschaftsfenster geben Sie die Dokumentklasse der Flash-Datei an.



Schritt 5

Die PictureGallery3D-Klasse

Sprite

Die Sprite-Klasse ist ein Grundbaustein der Anzeigelinste, der Grafiken anzeigen und auch untergeordnete Objekte enthalten kann. Ein Sprite-Objekt ähnelt einem Movieclip, besitzt aber keine Zeitleiste. Sprite ist die angemessene Basisklasse für Objekte, die keine Zeitleiste benötigen.

Öffnen Sie die Datei PICTUREGALLERY3D.AS mit Ihrem bevorzugten Editor, und fügen Sie folgenden Quelltext ein:

```
1: package{
2:     import flash.display.MovieClip;
3:     import flash.events.*;
4:     public class PictureGallery3D extends MovieClip
5:     {
6:         public function PictureGallery3D():void
7:         {
8:         }
9:     }
10: }
```

Die Dokumentklasse muss von `MovieClip` oder `Sprite` erben. Dazu importieren Sie in Zeile 2 die `MovieClip`-Klasse. In Zeile 3 importieren Sie zusätzlich das `flash.events`-Package, da Sie später auf das `ENTER_FRAME`-Event lauschen und Mausklicks auf die Bilder abfangen werden. Dazu aber später mehr. Erst einmal kümmern wir uns um das Laden der Bilder.

Schritt 6

Picture-Klasse anlegen

Ziel ist es, externe Bilder zu laden und diese später im dreidimensionalen Raum zu verteilen. Für die Bilder werden Sie eine eigene Klasse programmieren, die die Ladefunktionalität übernimmt. Erstellen Sie zu diesem Zweck mit dem Tastaturbefehl `[Strg]/[⌘]+[N]` eine neue ACTIONSCRIPT-DATEI, nennen Sie diese PICTURE.AS, und speichern Sie die Klasse im Projektordner.

Damit Sie später eine Instanz der Picture-Klasse auf die Bühne holen können, muss die Picture-Klasse von `Sprite` oder `MovieClip` erben. In diesem Fall reicht `Sprite` vollkommen aus, da Sie nicht alle Eigenschaften der `MovieClip`-Klasse benötigen werden. Da die `Sprite`-Klasse nicht so umfangreich ist wie die `MovieClip`-Klasse, werden Sie somit außerdem Speicher sparen. Öffnen Sie nun die Picture-Klasse, und übernehmen Sie folgenden Quelltext, um das Klassengerüst zu programmieren:

```
1: package{
2:     import flash.display.Sprite;
3:     public class Picture extends Sprite
4:     {
5:         public function Picture():void
6:         {
7:         }
8:     }
9: }
```

Um eine externe Datei zu laden, benötigen Sie die Loader-Klasse, die sich im Package `flash.display` befindet. Zusätzlich benötigen Sie die URLRequest-Klasse um, wie der Name schon sagt, eine URL-Anfrage zu erzeugen. Die URLRequest-Klasse befindet sich im Package `flash.net`. Importieren Sie also diese Klassen unter der ersten Import-Anweisung in der Picture-Klasse:

```
import flash.display.Loader;
import flash.net.URLRequest;
import flash.events.*;
```

In der letzten Zeile importieren Sie außerdem das `flash.events`-Package, um der Loader-Instanz später einen EventListener hinzufügen zu können, der Sie über den Ladevorgang informiert.

Da es sich bei den Dateien, die Sie laden wollen, um Bitmaps handelt und Sie später noch auf deren Eigenschaften zugreifen, müssen Sie außerdem die Bitmap-Klasse importieren. Ergänzen Sie unter dem eben eingefügten Code folgende Anweisung:

```
import flash.display.Bitmap;
```

Die Picture-Klasse soll am besten gleich bei der Instantiierung ohne weitere Methodenaufrufe von außen ein Bild laden. Deshalb schreiben Sie den Ladeaufruf in den Konstruktor. Der Konstruktor wird eine URL zu dem Bild in Form eines Strings erwarten. Später soll so das Laden eines Bildes ganz einfach mit dem Aufruf `new Picture("url/bild.jpg")` erledigt sein.

Schritt 7

Loader-Klasse importieren

Schritt 8

Bitmap-Klasse importieren

Schritt 9

Laden der Bilder

Exkurs zur

3D-Programmierung

Eignen Sie sich die Grundlagen der 3D-Programmierung in Flash im Exkurs auf Seite 184 an.

Schreiben Sie zu diesem Zweck den Konstruktor folgendermaßen um:

```
1: public function Picture(url:String):void
2: {
3:     var urlRequest:URLRequest = new URLRequest(url);
4:     var loader:Loader = new Loader();
5:     loader.load(urlRequest);
6:     addChild(loader);
7: }
```

In Zeile 3 wird ein neuer `URLRequest` mit der dem Konstruktor übergebenen URL gestartet und in der Variablen `urlRequest` gespeichert. Daraufhin erzeugen Sie eine Instanz der `Loader`-Klasse, deren `load`-Methode Sie `urlRequest` übergeben. Das Laden der Datei wird damit eingeleitet. In Zeile 6 holen Sie den `Loader` und damit die geladene Datei mit `addChild()` auf die Bühne. Dies soll erst einmal nur zum Testen dienen. In den nächsten Schritten werden wir das noch etwas abändern. Dazu kommen wir später.

Schritt 10

Testen der Picture-Klasse

Bilder von der Festplatte laden

Sie können natürlich auch Bilder von Ihrer Festplatte laden. Kopieren Sie Ihre Bilder dazu am besten in einen Unterordner `IMAGES` in den Projektordner. Übergeben Sie dann dem `Picture`-Konstruktor den relativen Pfad `"images/image1.jpg"`.

Damit Sie jetzt schon einmal testen können, ob die `Picture`-Klasse ihre Dienste erfüllt und eine Datei lädt, wechseln Sie nun zu der Flash-Datei. Klicken Sie in das erste Bild der ersten Ebene in der Hauptzeitleiste, und öffnen Sie das `ActionScript`-Fenster. Geben Sie folgenden Code ein:

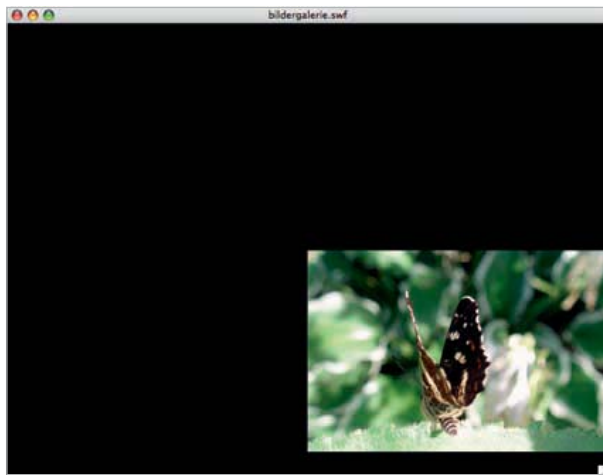
```
1: var picture:Picture = new Picture("http://flashfastforward.de/images/image1.jpg");
2: picture.x = stage.stageWidth/2;
3: picture.y = stage.stageHeight/2;
4: addChild(picture);
```

Sie erzeugen eine Instanz der `Picture`-Klasse, deren Konstruktor Sie eine URL zu einer Bilddatei übergeben. Anschließend positionieren Sie das `Picture`-Sprite auf die Bühnenmitte und fügen die Instanz der Bühne hinzu.

Schritt 11

Film testen

Testen Sie nun Ihren Flash-Film mit `Strg`/`⌘` + `↵`. Nach einer kurzen Ladezeit erscheint ein Bild auf der Bühne. Die `Picture`-Klasse funktioniert also. Das geladene Bild positioniert sich mit der oberen linken Ecke im Nullpunkt des `Picture`-Sprites. In den nächsten Schritten werden Sie das Bild mittig im `Picture`-Sprite positionieren.



◀ **Abbildung 3**
Das erste Bild wird geladen.

Wie Sie bereits in dem vorherigen 3D-Workshop gelernt haben, ist es sehr nützlich, den Registrierungspunkt eines Objekts, das im dreidimensionalen Raum positioniert werden soll, mittig anzuordnen. Da das Bild jedoch dynamisch in das Picture-Sprite geladen wird, befindet sich dessen Registrierungspunkt automatisch oben links im Nullpunkt. Um dies zu ändern, müssen Sie das Bild neu positionieren.

Sie haben dabei erst Zugriff auf die Eigenschaften der geladenen Datei, wenn der Ladevorgang abgeschlossen ist. Deswegen müssen Sie auf das `INIT`-Event der Loader-Klasse lauschen, das genau dann gesendet wird. Zusätzlich soll das Bild auch erst beim Senden dieses Events der Bühne hinzugefügt werden.

Ersetzen Sie zu diesem Zweck die Zeile Ihrer Picture-Klasse, in der Sie bereits `addChild()` aufgerufen haben (siehe Schritt 9, Zeile 6). Schreiben Sie stattdessen folgende Codezeile, die dazu dient, beim `INIT`-Event eine Methode `initListener()` aufzurufen:

```
loader.contentLoaderInfo.addEventListener(Event.INIT,  
    initListener);
```

In der Methode `initListener()` werden Sie nun das Bild neu positionieren und der Bühne hinzufügen.

`initListener()` wird aufgerufen, wenn die geladene Datei initialisierbar ist. Schreiben Sie die Methode folgendermaßen unterhalb des Konstruktors:

Schritt 12

Hinzufügen eines LoaderListeners

Schritt 13

Bild mittig positionieren

```

1: private function initListener(event:Event):void
2: {
3:     var bmp:Bitmap = event.target.content;
4:     bmp.x = -bmp.width/2;
5:     bmp.y = -bmp.height/2;
6:     addChild(bmp);
7: }

```

In Zeile 3 speichern Sie das geladene Bild in einer Variablen `bmp`, um es über diese referenzieren zu können. In den nächsten zwei Zeilen setzen Sie die x- und y-Position des geladenen Bildes auf die Hälfte von dessen Breite bzw. Höhe. Das Bild wird damit im Picture-Sprite mittig positioniert. In Zeile 6 fügen Sie nun das Bild per `addChild()` der Bühne hinzu.

Schritt 14

Film testen

Testen Sie nun Ihren Flash-Film mit `Strg`/`⌘` + `↵`. Wie Sie sehen, befindet sich das Bild nun in der Mitte der Bühne bzw. in der Mitte des Picture-Sprite.

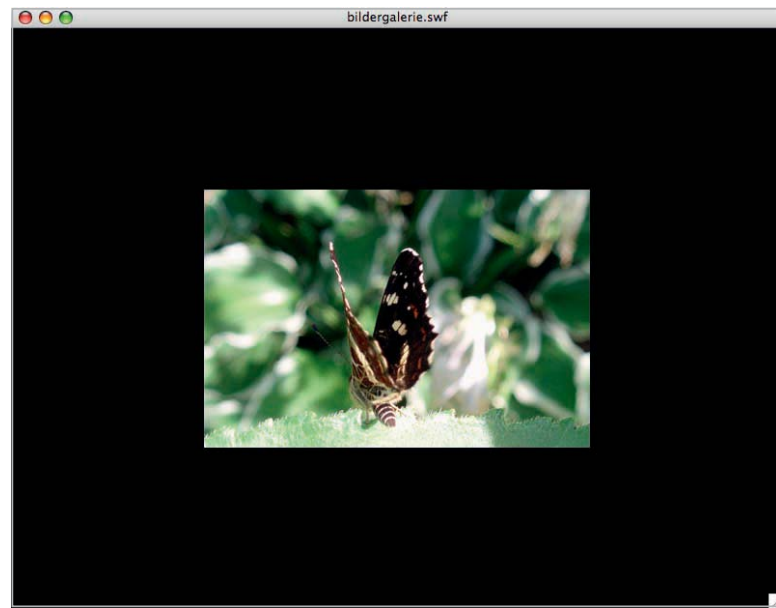


Abbildung 4 ►

Das Bild wird geladen und mittig positioniert.

Schritt 15

Skript auf der Hauptzeit- leiste löschen

Wichtig ist, dass Sie nun das Skript auf der Hauptzeitleiste wieder löschen. Es diente nur zum Testen der Picture-Klasse.

Die eigentliche Funktionalität Ihres Programms wird nun in der Dokumentklasse `PictureGallery3D` programmiert.

Wechseln Sie nun zu der `PictureGallery3D`-Klasse.

Sie benötigen für einfache 3D-Anwendungen nur wenige Variablen, mit denen Sie später, mit Hilfe der Formel zur Umrechnung der 3D-Position, die tatsächliche Position im zweidimensionalen Koordinatensystem von Flash errechnen.

Deklarieren Sie nun die Variable `focalLength` sowie die Aussichtspunkt-Variablen im Deklarationsbereich über dem Konstruktor:

```
1: private var focalLength:int = 250;
2: private var viewPointX:int = stage.stageWidth/2;
3: private var viewPointY:int = stage.stageHeight/2;
4: private var distanceZ:int = 300;
```

In Zeile 1 deklarieren Sie die Variable für die Brennweite, `focalLength`, und initialisieren sie mit dem Wert 250. Als Nächstes legen Sie die Koordinaten des Aussichtspunkts in den Variablen `viewPointX` und `viewPointY` auf die Bühnenmitte. Zusätzlich deklarieren Sie in Zeile 4 eine Variable `distanceZ` und weisen dieser den Wert 300 zu. `distanceZ` wird den Abstand zwischen den Bildern auf der z-Achse bestimmen. Deklarieren Sie des Weiteren folgende Variablen:

```
5: private var pictures:Array;
```

In dem Array `pictures` werden Referenzen auf die Picture-Sprites gehalten, um später die Tiefe der Sprites nach ihrer z-Position sortieren zu können. Dazu kommen wir aber erst später.

In diesem Schritt werden die Bilder geladen, die Sie in Ihre 3D-Bildergalerie aufnehmen. Erweitern Sie den Konstruktor der `PictureGallery3D`-Klasse mit folgendem Code:

```
1: var url:String = "http://www.flashfastforward.de/images/";
2: var urls:Array = new Array(url+"image1.jpg",url+"image2.
   jpg",url+"image3.jpg",url+"image4.jpg",url+"image5.jpg");
3: pictures = new Array();
4: for(var i:uint = 0; i< urls.length; i++){
5:     var picture:Picture = new Picture(urls[i]);
6:     addChild(picture);
7:     pictures.push(picture);
8: }
```

Schritt 16

Variablen deklarieren

Schritt 17

Bilder laden

Um gleich etwas Tipparbeit zu sparen, und da die Bilddateien von einem externen Server geladen werden, erzeugen Sie eine Variable `url` vom Typ `String`, in der Sie die URL des Bilder-Servers speichern. In der nächsten Zeile füllen Sie ein Array `urls` mit den vervollständigten URLs zu den Bildern.

Bevor Sie die Bilder laden, weisen Sie der Klassenvariablen `pictures` ein neues Array zu. In diesem sollen die Picture-Sprites gespeichert werden. In der `for`-Schleife ab Zeile 4 durchlaufen Sie das Array `urls` und erzeugen dabei neue Instanzen Ihrer Picture-Klasse, der Sie die in `urls` gespeicherte URL übergeben. Die Instanz speichern Sie in der Variablen `picture` und fügen diese der Bühne hinzu. Daraufhin speichern Sie diese im `pictures`-Array.

Schritt 18

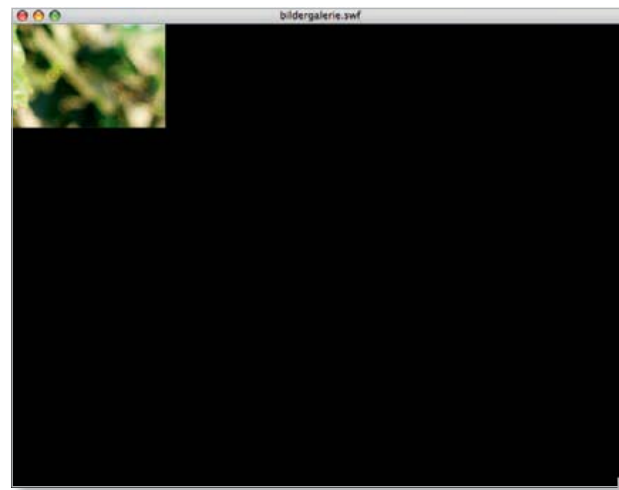
Film erneut testen

Elemente dynamisch positionieren

Um Elemente im 3D-Raum anzuzeigen, ist es besser, den Registrierungspunkt dieser Elemente mittig zu positionieren. Bilder zum Beispiel werden dann bei Position Null auf allen drei Achsen in voller Größe in der Mitte der Bühne angezeigt. Da Sie die Bilder jedoch dynamisch laden, befindet sich deren Registrierungspunkt automatisch oben links.

Testen Sie nun Ihren Flash-Film mit `Strg`/`⌘` + `↵`. Nach einer kurzen Ladezeit erscheinen die Bilder nacheinander in der linken oberen Ecke der Bühne.

Da die Bilder im Picture-Sprite mittig positioniert sind und dieser sich auf Position 0,0 der Bühne befindet, ist nur ein Ausschnitt der Bilder zu sehen.



▲ **Abbildung 5**

Die Bilder werden nach dem Laden im Sprite mittig positioniert. Das Sprite befindet dabei oben links.

Schritt 19

3D-Eigenschaften deklarieren

Jetzt müssen Sie die Eigenschaften für die x-, y- und z-Position im dreidimensionalen Raum deklarieren. Wechseln Sie zu der Picture-Klasse. Deklarieren Sie folgende Variablen im Deklarationsbereich über dem Konstruktor:


```
public var posZ:Number;  
public var posX:Number;  
public var posY:Number;  
public var speedZ:Number;
```

Zusätzlich zu den Positionsvariablen deklarieren Sie in der letzten Zeile eine Eigenschaft `speedZ`. Diese Variable soll die Geschwindigkeit bestimmen, mit der sich die Bilder aus dem endlosen Raum in Richtung Betrachter bewegen.

Die Positionierung und Steuerung der Bilder nehmen Sie in der Dokumentklasse `PictureGallery3D` vor. Wechseln Sie also wieder zu dieser Klasse. Erweitern Sie die `for`-Schleife im Konstruktor um folgende Zeilen:

```
1: picture.posZ = distanceZ;  
2: picture.posX = Math.random()*2000-1000;  
3: picture.posY = Math.random()*2000-1000;  
4: picture.speedZ = -1;  
5: distanceZ += 300;
```

In Zeile 1 setzen Sie die z-Position von `picture` auf den `distanceZ`-Wert. In den beiden nächsten Zeilen legen Sie die x- und y-Position auf einen zufälligen Wert zwischen 1000 und -1000. Da sich der Aussichtspunkt in der Mitte der Bühne befindet, werden die Bilder zufällig rund um die Bühnenmitte angeordnet. In Zeile 4 setzen Sie `speedZ` auf -1. In Zeile 5 addieren Sie 300 zu dem `distanceZ`-Wert.

Im nächsten Durchlauf der `for`-Schleife wird damit das nächste Bild umso weiter in den Raum hinein positioniert.

Die Position der Bilder muss permanent neu berechnet werden. Zu diesem Zweck fügen Sie dem `ENTER_FRAME`-Event der `PictureGallery3D`-Klasse einen `EventListener` hinzu, der beim Senden dieses Events eine Methode `enterFrameListener()` aufruft. Erledigen Sie dies in der letzten Zeile des Konstruktors (außerhalb der `for`-Schleife):

```
this.addEventListener(Event.ENTER_FRAME,enterFrameListener);
```

Schritt 20

3D-Eigenschaften initialisieren

Schritt 21

EventListener hinzufügen

Schritt 22

3D-Engine programmieren

Die Methode `enterFrameListener()` soll die 3D-Eigenschaften der `Picture-Sprites` errechnen. Programmieren Sie die Methode folgendermaßen unterhalb des Konstruktors:

```
1: private function enterFrameListener(event:Event):void
2: {
3:     for(var i:uint = 0; i < pictures.length; i++){
4:         var picture:Picture = Picture(pictures[i]);
5:         picture.posZ += picture.speedZ;
6:         var scale:Number = focalLength/(focalLength +
           picture.posZ);
7:         picture.scaleX = picture.scaleY = scale;
8:         picture.x = viewPointX + picture.posX * scale;
9:         picture.y = viewPointY + picture.posY * scale;
10:    }
11: }
```

3D in Flash

Mehr Informationen zu der Formel zur Berechnung der 3D-Eigenschaften können Sie dem 3D-Exkurs auf Seite 184 entnehmen.

In einer `for`-Schleife durchlaufen Sie das `pictures`-Array, das Referenzen auf die `Picture-Sprites` beinhaltet. In Zeile 4 verweisen Sie mit der Variablen `picture` auf diese Referenz, um diese so leichter ansprechen zu können. Darauf folgend addieren Sie zu der `z`-Position (`posZ`) von `picture` den im letzten Schritt auf `-1` festgelegten `speedZ`-Wert. `posZ` wird damit permanent verkleinert, und das Bild bewegt sich auf den Betrachter zu.

Von Zeile 6 bis 9 wenden Sie die 3D-Formel an, um die 3D-Eigenschaften auf die tatsächlichen Eigenschaften im zweidimensionalen Koordinatensystem umzurechnen.

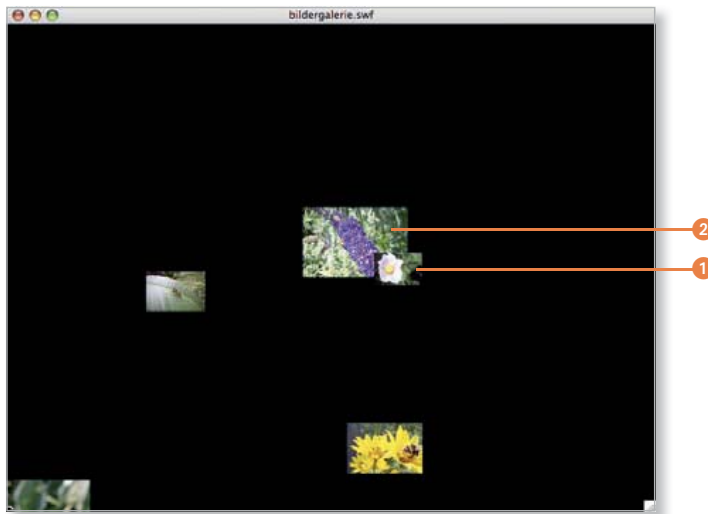
Schritt 23

Film testen

Testen Sie nun wieder Ihren Flash-Film mit `Strg`/`⌘` + `↵`. Wie Sie sehen, verteilen sich die Bilder im dreidimensionalen Raum und bewegen sich auf Sie zu.

Testen Sie den Film des Öffneren, werden Sie sehen, dass die Bilder zufällig positioniert werden.

Es könnte allerdings sein, dass ein Bild ein anderes überdeckt, obwohl dieses eigentlich weiter hinten im Raum sein sollte als das überdeckte Bild. In Abbildung 6 auf der nächsten Seite verdeckt das Bild ① ein anderes Bild ②, das eigentlich weiter vorn liegt und somit hinter Bild ① liegen müsste. Dem müssen Sie Abhilfe leisten, indem Sie die Bilder entsprechend ihrer `z`-Position sortieren.



◀ Abbildung 6

Die Bilder bewegen sich bereits auf den Betrachter zu, allerdings überlappen sich einige Bilder nicht so, wie es sein sollte.

Nun ist es soweit, dass Sie die in dem 3D-Exkurs erklärte Formel zum Sortieren der Tiefe anwenden. Schreiben Sie die Methode `sortZ()` folgendermaßen unterhalb der `enterFrameListener`-Methode:

```
private function sortZ():void
{
    pictures.sortOn("posZ", Array.DECENDING | Array.NUMERIC);
    for(var i:uint = 0; i < pictures.length; i++){
        var picture:Picture = pictures[i];
        setChildIndex(picture,i);
    }
}
```

Sie müssen diese Methode nun nur noch in `enterFrameListener()` aufrufen – und alles Weitere geschieht von selbst. Erledigen Sie dies in der letzten Zeile der `enterFrameListener`-Methode mit folgendem Aufruf:

```
sortZ();
```

Testen Sie nun Ihren Flash-Film mit `Strg`/`⌘` + `↵`. Wenn sich vorher Bilder nicht entsprechend der z-Position überlagert haben, ist nun die Darstellung korrigiert.

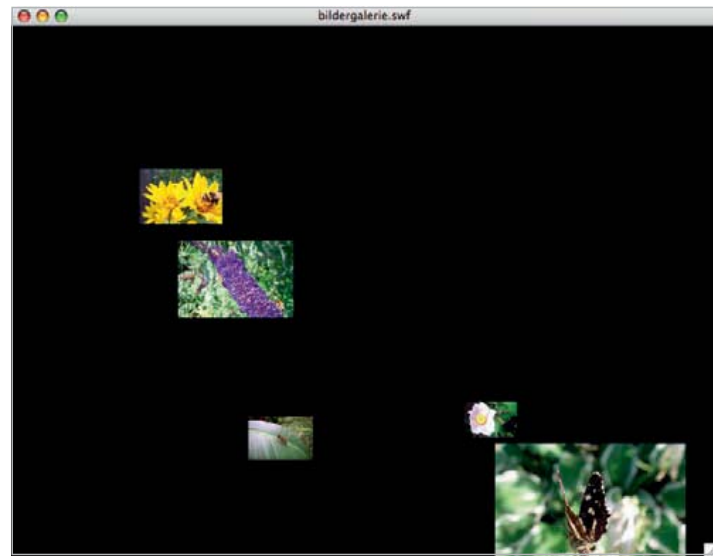
Schritt 24

Z-Sortierung

Schritt 25

Film testen

Abbildung 7 ►
Die z-Sortierung der Bilder
im Raum stimmt.



Schritt 26

Noch mehr Bilder

Mit dem Ergebnis können Sie im Grunde schon recht zufrieden sein. Allerdings wird momentan der 3D-Effekt noch nicht ganz deutlich, da nur fünf Bilder durch den Raum schweben. Aus diesem Grunde laden Sie jetzt weitere Bilder in Ihre Bildergalerie. Schreiben Sie im Konstruktor folgende Zeilen unter die Initialisierung von `urls`:

```
urls.push(url+"image6.jpg",url+"image7.jpg",url+"image8.jpg",  
url+"image9.jpg",url+"image10.jpg");  
urls = urls.concat(urls);
```

In der ersten Zeile nehmen Sie in das Array `urls` noch weitere fünf Bilder auf. Weil zehn Bilder immer noch nicht wirklich reichen, verketteten Sie `urls` einfach mit sich selbst. Damit wird der gesamte Inhalt noch einmal an `urls` angehängt. Sie laden somit alle Bilder von `urls` einfach ein zweites Mal.

Später, wenn Sie den Workshop beendet haben, können Sie stattdessen URLs zu Ihren eigenen Bildern hinzufügen.

Schritt 27

Film testen

Testen Sie nun wieder Ihren Flash-Film mit `Strg`/`⌘` + `↩`. Die Bilder reichen weit in den Raum hinein und bewegen sich langsam auf Sie zu. Dieses Ergebnis kann sich sehen lassen, oder?

Wenn Sie den Film eine Weile laufen lassen, werden Sie feststellen, dass Bilder, die an Ihnen vorbeigeflogen sind, gespiegelt in die falsche Richtung weiterfliegen ❶, ❷ und ❸. Das gilt es im nächsten Schritt zu verhindern.



◀ **Abbildung 8**

Wenn die Bilder an Ihnen vorbeifliegen, wechseln sie die Richtung und fliegen gespiegelt davon.

Das im letzten Schritt beschriebene Problem ist ganz einfach zu lösen. Schreiben Sie die Codezeilen 2 bis 7 oberhalb der Berechnung des `scale`-Wertes in der `enterFrameListener`-Methode:

```
1: picture.posZ....
2: if(picture.posZ < -focalLength){
3:     picture.posZ += distanceZ;
4: }
5: else if(picture.posZ > distanceZ - focalLength){
6:     picture.posZ -= distanceZ;
7: }
8: var scale:Number = ...
```

In der ersten `if`-Abfrage prüfen Sie, ob die `z`-Position des Bildes kleiner ist als der negative Wert von `focalLength`, also kleiner als `-250`. Ist dies der Fall, würde das Bild »hinter« dem Betrachter negativ skaliert werden und in die falsche Richtung fliegen. Deshalb addieren Sie den `distanceZ`-Wert zur `z`-Position des Bildes. Das Bild wird somit an den entferntesten Punkt auf der `z`-Achse gesetzt.

Bedenken Sie: Sie haben die einzelnen Bilder immer 300 Pixel weiter nach hinten versetzt und die Gesamtdistanz in der `distanceZ`-Variablen gespeichert. Raffiniert! Das Bild wird sich also wieder in die Reihe ganz nach hinten einordnen. Die zweite `if`-Abfrage erledigt genau dasselbe, nur in die andere Richtung. Wenn das Bild über den `distanceZ`-Wert hinausfliegt, wird es ganz nach vorne, direkt vor den Betrachter gesetzt. Sie können später `speedZ` einen positiven Wert zuweisen, um dies zu testen. Dann würden die Bilder nach hinten in den Raum fliegen.

Schritt 28

3D-Engine erweitern

Schritt 29

Film testen

Kurze Pause?

Wenn Ihnen dieser Workshop bisher viel Mühe gemacht hat, können Sie die nächsten Schritte auch überspringen, damit Sie am Ende nicht noch Kopfschmerzen bekommen. Das Ergebnis lässt sich ja auch schon sehen!

Um die Engine zu testen, können Sie den `speedZ`-Wert im Konstruktor etwas verkleinern. Dann sehen Sie den Effekt der `if`-Abfragen des vorherigen Schritts schneller. Ein Wert von `-10` sollte ausreichen.

Testen Sie nun Ihren Flash-Film. Wie Sie sehen, fliegen die Bilder auf Sie zu. Sobald die Bilder scheinbar hinter Ihnen verschwinden müssten, tauchen sie wieder »am Horizont« auf. Vergessen Sie nach dem Testen nicht, den `speedZ`-Wert wieder auf `-1` zu setzen, da die Bilder ansonsten weiterhin so schnell durch den Raum fliegen würden.

Sie haben damit eine vollwertige Engine für eine 3D-Bildergalerie programmiert. Aber das reicht uns natürlich nicht! In den nächsten Schritten werden Sie die Bilder anklickbar machen, damit Sie auf die angeklickten Bilder zufliegen können. (In Wahrheit fliegen alle Bilder gleichzeitig auf Sie zu, aber das muss man ja keinem verraten.)

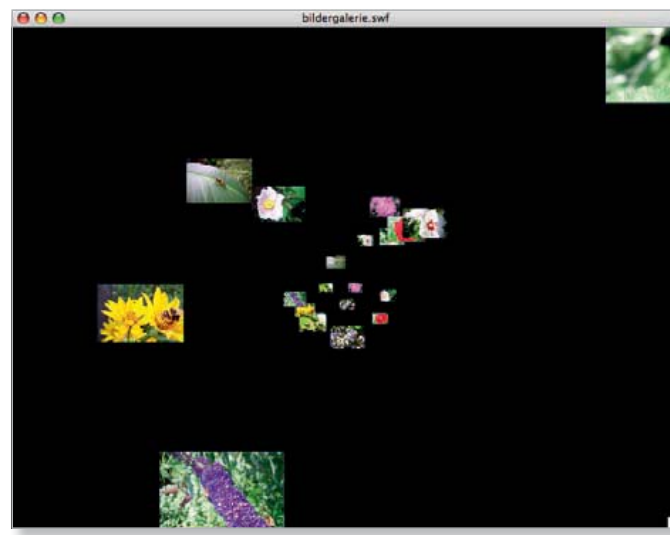


Abbildung 9 ►

Mehr Bilder für einen beeindruckenderen Effekt: Die Bilder bewegen sich endlos im 3D-Raum.

Schritt 30

Klick auf die Bilder abfangen

Wechseln Sie wieder zu der `PictureGallery3D`-Klasse. Um einen Klick abzufangen, verwenden Sie einen Maus-Listener. Fügen Sie zu diesem Zweck im Konstruktor in das Ende der `for`-Schleife, die die `Picture`-Sprites initialisiert, folgende Codezeile ein:

```
picture.addEventListener(MouseEvent.CLICK, mouseClickListener);
```

Hierdurch wird die Methode `mouseClickListener()` bei einem Klick auf eines der Bilder aufgerufen. Diese Methode programmieren Sie im nächsten Schritt.

Um aus allen Methoden der Klasse abfragen zu können, welches Bild angeklickt wurde, benötigen Sie eine Klassenvariable, die eine Referenz auf das angeklickte Bild speichert. Deklarieren Sie dazu im Deklarationsbereich der Klasse eine Variable `currentPicture` des Typs `Picture`:

```
private var currentPicture:Picture;
```

Programmieren Sie nun die Methode `mouseDownListener()` unter der letzten Methode:

```
1: private function mouseDownListener(event:Event):void
2: {
3:     currentPicture = Picture(event.target);
4: }
```

In Zeile 3 speichern Sie das `target` des Event-Objekts, wobei es sich um das angeklickte Bild handelt, in `currentPicture`.

Jetzt müssen Sie die `enterFrameListener`-Methode anpassen, um den Kamerafahrt-Effekt zu erzeugen. In der Methode fragen Sie ab, auf welches Bild `currentPicture` verweist, um dann die Positionen aller Bilder so anzupassen, dass das angeklickte Bild in der Mitte und in voller Größe zu sehen ist. Ersetzen Sie dafür folgende Zeile der `for`-Schleife in der `enterFrameListener`-Methode

```
picture.posZ += picture.speedZ;
```

durch folgenden Code:

```
1: if(currentPicture != null){
2:     picture.posZ+=(0-currentPicture.posZ)/10;
3:     picture.posX+=(0-currentPicture.posX)/10;
4:     picture.posY+=(0-currentPicture.posY)/10;
5: }
6: else{
7:     picture.posZ += picture.speedZ;
8: }
```

In Zeile 1 fragen Sie ab, ob die Klassenvariable `currentPicture` überhaupt gesetzt ist. Ist dies nicht der Fall, wurde auch kein Bild angeklickt. Der

Schritt 31

Angeklicktes Bild zwischenspeichern

Schritt 32

Kamerafahrt programmieren

`else`-Rumpf ab Zeile 6 bestimmt in diesem Fall, alles seinen gewohnten Verlauf nehmen zu lassen, d.h., die Bilder sollen, angetrieben durch `speedZ`, durch den Raum fliegen.

Wenn ein Bild angeklickt wurde, errechnen Sie ab Zeile 2 die Differenz der Position von `currentPicture` zur Nullposition aller drei Achsen und gleichen die Differenz langsam Null an. Den so errechneten Wert addieren Sie auf die Position von jedem Bild, wodurch sich alle Bilder gleichmäßig bewegen, bis das angeklickte Bild den Nullpunkt erreicht hat.

Schritt 33 Film testen

Testen Sie nun wieder Ihren Flash-Film. Die Bilder fliegen Ihnen wie gewohnt entgegen. Klicken Sie jetzt am besten auf eines der hinteren Bilder, um den Effekt bewundern zu können. Die Bilder fliegen durch den Raum! Die Kamerafahrt funktioniert.

Nach einem Klick auf ein Bild ist es jedoch so, dass `currentPicture` immer gesetzt ist. Das heißt, dass die normale Kamerafahrt nicht wieder einsetzt. Sie können zwar ein anderes Bild anklicken und anfahren, jedoch kehren Sie nicht zur normalen Kamerafahrt zurück. Dem werden wir in den nächsten Schritten Abhilfe verschaffen.

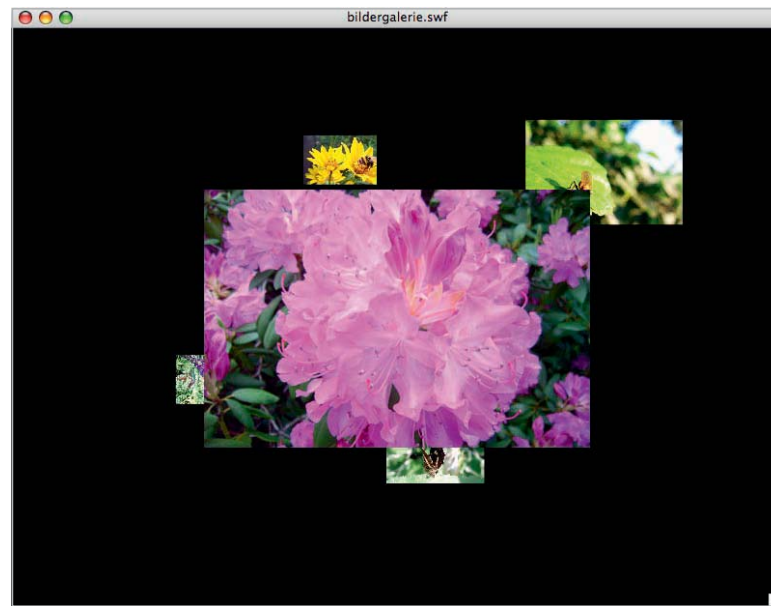


Abbildung 10 ►

Die Kamerafahrt funktioniert und fährt auf das angeklickte Bild zu.

Schritt 34 Startposition merken

Damit Sie die Kamera so herausfahren können, dass die Bilder wieder ihre ursprünglichen x- und y-Positionen einnehmen, benötigen Sie zwei weitere Klassenvariablen, in denen Sie die Anfangsposition des Bildes spei-

chern. Wechseln Sie also zu der Picture-Klasse, und deklarieren Sie die beiden folgenden Variablen:

```
public var startX:Number;  
public var startY:Number;
```

Wechseln Sie nun wieder in die PictureGallery3D-Klasse, und ändern Sie die Initialisierung der Picture-Sprite-Eigenschaften der `for`-Schleife im Konstruktor:

```
picture.posX = Math.random()*2000-1000;  
picture.posY = Math.random()*2000-1000;
```

Setzen Sie stattdessen folgenden Code ein:

```
picture.posX = picture.startX = Math.random()*2000-1000;  
picture.posY = picture.startY = Math.random()*2000-1000;
```

Damit setzen Sie sowohl die `posX`- und `posY`-Eigenschaften als auch die soeben deklarierten `startX`- und `startY`-Eigenschaften von `picture` auf dieselben Werte.

Die Kamera soll wieder herausfahren, wenn man ein zweites Mal auf das bereits angeklickte Bild klickt. Deklarieren Sie zu diesem Zweck eine boolesche Variable, die auf `true` gesetzt wird, wenn der zweite Klick erfolgt. Diese Variable soll so lange `true` bleiben, bis die x- und y-Position der Bilder wieder dem Ausgangspunkt entspricht. Nennen Sie die Variable `isZoomOut` und fügen Sie sie im Deklarationsbereich der Klasse ein:

```
private var isZoomOut:Boolean = false;
```

`isZoomOut` erhält gleich von Anfang an den Wert `false`, da beim Start des Films kein Bild angeklickt ist. Ändern Sie nun die in Schritt 32 programmierte `if`-Abfrage der `enterFrameListener`-Methode folgendermaßen:

```
1: if(currentPicture != null && !isZoomOut){  
2:     picture.posZ+=(0-currentPicture.posZ)/10;  
3:     picture.posX+=(0-currentPicture.posX)/10;  
4:     picture.posY+=(0-currentPicture.posY)/10;
```

Schritt 35

Kamera herausfahren

```

5:  }
6:  else if(isZoomOut){
7:      picture.posZ+=(100-currentPicture.posZ)/10;
8:      picture.posX+=(currentPicture.startX-currentPicture.
        posX)/10;
9:      picture.posY+=(currentPicture.startY-currentPicture.
        posY)/10;
10:     if(int(currentPicture.posX) == int(currentPicture.startX))
        {
11:         currentPicture = null;
12:         isZoomOut = false;
13:     }
14: }
15: else{
16:     picture.posZ += picture.speedZ;
17: }

```

In Zeile 1 fragen Sie nun ab, ob `currentPicture` gesetzt ist und ob `isZoomOut` den Wert `false` hat. Es soll also nur in das Bild hineingefahren werden, wenn nicht gerade herausgefahren wird – logisch!

In Zeile 6 fragen Sie ab, ob `isZoomOut` gleich `true` ist, ob also die Kamera herausgefahren werden soll. In diesem Fall wenden Sie dasselbe Prinzip wie in Schritt 32 an. Diesmal geben Sie als Zielposition jedoch die gespeicherten `startX`- und `startY`-Positionen von `currentPicture` an. Die `z`-Position lassen Sie um 100 Schritte zurückrechnen, damit die Bilder wieder ein wenig in den Raum zurückfahren.

Nach der beendeten Kamerafahrt müssen Sie `isZoomOut` wieder auf `false` setzen, damit die Bilder wieder durch `speedZ` angetrieben werden. Außerdem müssen Sie den Wert von `currentPicture` auf `null` setzen, da ansonsten die `if`-Bedingung in Zeile 3 erfüllt bliebe und die Kamera wieder auf das Bild zufahren würde. Das alles erledigen Sie in den Zeilen 10 bis 13, in denen Sie abfragen, ob `posX` und `startX` von `currentPicture` gleich sind. Damit der Vergleich weniger fehleranfällig ist, wandeln Sie die Werte zuerst in runde Integerwerte um.

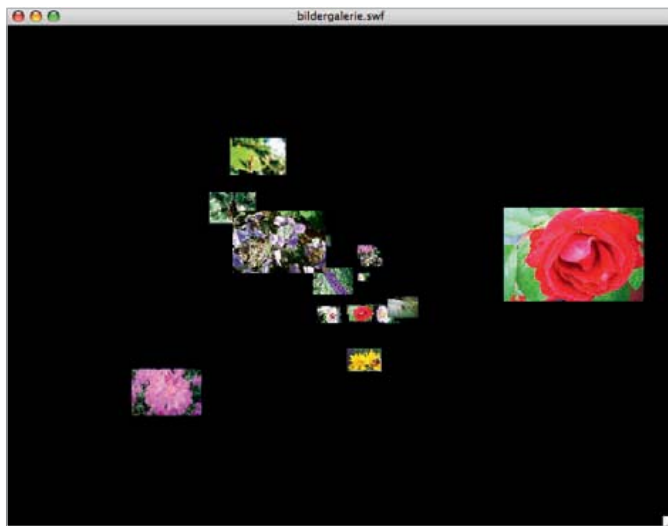
Schlussendlich, in Zeile 15, legen Sie in der `else`-Bedingung fest, dass alle Bilder, für den Fall, dass die Kamera weder herein- noch herausfährt, mit der Geschwindigkeit `speedZ` durch den Raum fliegen.

Eine Sache fehlt allerdings noch, damit das Ganze funktioniert: Sie müssen den Wert von `isZoomOut` beim Klick auf ein Bild setzen. Erweitern Sie die Methode `mouseDownListener()` um folgende erste Zeile:

```
isZoomOut = currentPicture == Picture(event.target);
```

Mit dieser Zeile prüfen Sie, ob `currentPicture` dem angeklickten Bild entspricht. Je nachdem, wird `isZoomOut` auf `true` oder `false` gesetzt.

Testen Sie nun wieder Ihren Flash-Film. Die Bilder fliegen Ihnen wie gewohnt entgegen. Wenn Sie nun ein Bild anklicken, fährt die Kamera auf dieses zu. Klicken Sie ein zweites Mal auf das Bild, fährt die Kamera zurück.



Sie können sich auf die Schulter klopfen: Sie haben nicht nur ein 3D-Programm geschrieben, sondern damit auch eine sehr schöne Bildergalerie erschaffen, die Sie schleunigst auf Ihrer Homepage veröffentlichen sollten.

Schritt 36

Ergebnis testen

◀ Abbildung 11

Die Bildergalerie ist fertig. Ein hartes Stück Arbeit, das sich gelohnt hat!

3D-Karussellmenü

Menüelemente im dreidimensionalen Raum

```
package{
    import flash.display.*;
    import flash.events.*;
    public class CarouselMenu extends MovieClip
    {
        private var focalLength:int = 250;
        private var viewPointX:int = stage.stageWidth/2;
        private var viewPointY:int = stage.stageHeight/3;
        private var radius:uint = 400;
        private var offsetZ:uint = 400;
        private var numItems:uint = 10;
        private var items:Array;
        private var currentItem:CarouselItem;
        public function CarouselMenu():void
        {
            items = new Array();
            for (var i : uint = 0; i < numItems; i ++)
```

In diesem Workshop werden Sie ein Menü der besonderen Art erstellen. Ziel ist es, die Menüelemente im dreidimensionalen Raum in Form eines Kreises zu verteilen und ein Karussell daraus zu erzeugen. Mit der Maus können Sie das Karussell drehen und so ein angesteuertes Element nach vorne bringen. Wenn Sie die 3D-Engine erst einmal fertig haben, können Sie Ihrer Fantasie freien Lauf lassen und das Menü zum Beispiel zu einer Bildergalerie umfunktionieren.



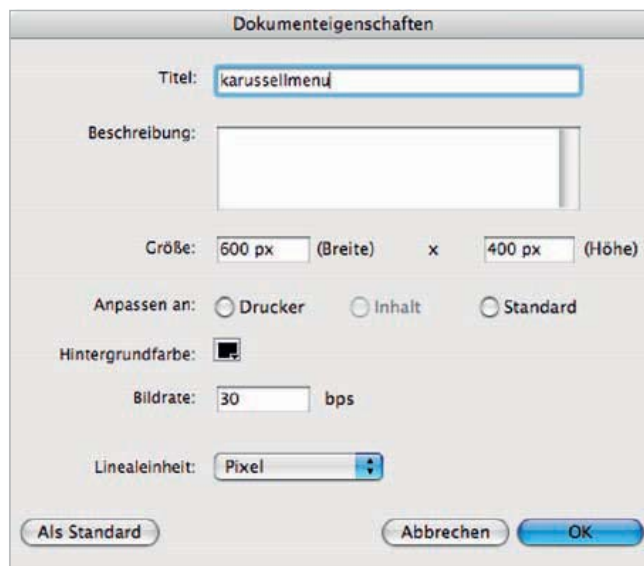
Zielsetzungen:

- Abgerundetes Rechteck programmieren
- 3D-Welt erschaffen
- Menüelemente im Karussell anordnen
- Karussell drehen lassen
- Menüelemente anklickbar machen

[Ergebnis: 20_3D-KARUSSELL-MENUE • CAROUSELMENU.SWF]

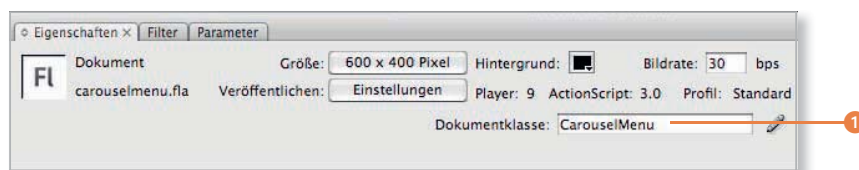
Im ersten Schritt legen Sie einen neuen Ordner auf Ihrer Festplatte an und nennen ihn 3D-KARUSSELLMENUE. Hier werden Sie die von Ihnen erstellten Dateien zu diesem Workshop abspeichern.

Erstellen Sie ein neues AS3-Flash-Dokument mit einer Größe von 600x400 Pixel und einer BILDRADE von 30bps. Stellen Sie die HINTERGRUNDFARBE auf Schwarz. Speichern Sie die Datei unter dem Namen KARUSSELLMENU.FLA im Projektordner ab.



Eine Dokumentklasse wird in diesem Workshop die allgemeine Steuerung der Flash-Datei übernehmen. Erstellen Sie zu diesem Zweck mit dem Tastaturbefehl **[Strg]/[⌘]+[N]** eine neue ACTIONSCRIPT-DATEI, nennen Sie diese CAROUSELMENU.AS, und speichern Sie die Klasse im Projektordner ab.

Jetzt können Sie im Eigenschaftenfenster, das Sie über **[Strg]/[⌘]+[F3]** einblenden, die DOKUMENTKLASSE »CarouselMenu« **1** eintragen. Bestätigen Sie die Eingabe mit **[↵]**.



Schritt 1 Vorbereitungen

Schritt 2 Flash-Dokument erstellen

◀ **Abbildung 1**
Dokumenteigenschaften festlegen

Dokumentklasse?

Lesen Sie im Workshop »3D-Sternenfeuer« auf Seite 188 nach, wie Dokumentklassen funktionieren und wie sie eingesetzt werden.

Schritt 3 Dokumentklasse anlegen

Schritt 4 Dokumentklasse festlegen

◀ **Abbildung 2**
Klicken Sie auf eine leere Stelle auf der Bühne, um sich die Dokumenteigenschaften im Eigenschaftenfenster anzeigen zu lassen.

Schritt 5 CarouselItem-Klasse anlegen

Ziel ist es jetzt, eine Klasse zu programmieren, die die Eigenschaften der Karussell-Menüelemente bestimmt. Mit einer CarouselItem-Klasse werden Sie verschiedenfarbige Rechtecke erzeugen, die als Menüpunkte fungieren.

Erstellen Sie zu diesem Zweck mit dem Tastaturbefehl `Strg/⌘ + N` eine neue ACTIONSCRIPT-DATEI, und speichern Sie die Klasse unter dem Namen CAROUSELITEM.AS im Projektordner.

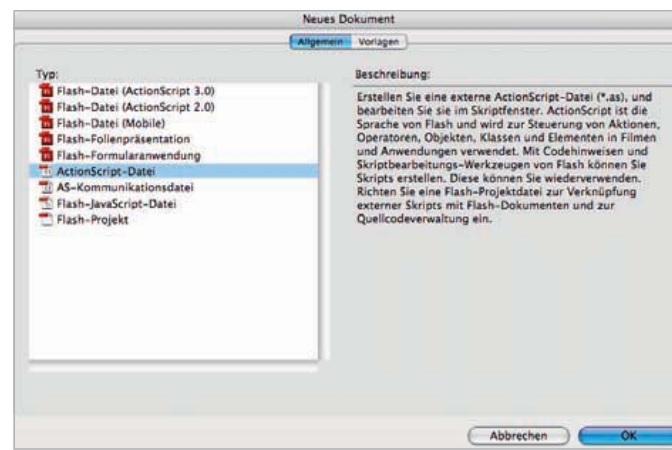


Abbildung 3 ►
Eine neue ActionScript-Datei
wird angelegt.

Schritt 6 Die CarouselItem-Klasse

Damit Sie später eine Instanz der CarouselItem-Klasse der Bühne hinzufügen können, muss die Klasse von `Sprite` oder `MovieClip` erben. In diesem Fall reicht `Sprite` vollkommen aus, da nicht alle Eigenschaften der `MovieClip`-Klasse verwendet werden.

Öffnen Sie nun die CarouselItem-Klasse und übernehmen Sie folgenden Quelltext:

```
1: package {  
2:     import flash.display.*;  
3:     public class CarouselItem extends Sprite {  
4:         public function CarouselItem():void  
5:         {  
6:         }  
7:     }  
8: }
```

Dies ist das Standardgerüst der Klasse.

Wie in Schritt 5 beschrieben, soll ein Karussellelement aus einem dynamisch gezeichneten Rechteck bestehen. Um dieses per Code zu erzeugen, schreiben Sie folgenden Quelltext in den Konstruktor:

```
var colorRect:Shape = new Shape ();
colorRect.graphics.lineStyle(5,0xffffffff);
colorRect.graphics.beginFill(Math.random()* 0xffffffff);
colorRect.graphics.drawRoundRect(-50, -50, 100, 100, 10);
colorRect.graphics.endFill();
addChild(colorRect);
```

Um das abgerundete Rechteck zu programmieren, benutzen Sie die Shape-Klasse. In der zweiten Zeile setzen Sie die Rahmenfarbe auf Weiß, mit einer Stärke von 5 Pixel. In der nächsten Zeile bestimmen Sie die Füllfarbe, die Sie auf einen zufälligen Farbwert setzen.

Danach verwenden Sie die `drawRoundRect`-Methode, um das abgerundete Rechteck zu erzeugen. Die x- und y-Position setzen Sie jeweils auf -50 Pixel. Die Breite und Höhe legen Sie mit 100 Pixel fest. Dadurch zentrieren Sie das Rechteck im `CarouselItem`-Sprite. Der vierte Parameter der `drawRoundRect`-Methode bestimmt den Rundungswinkel. Diesen setzen Sie auf 10. Danach fügen Sie das Rechteck der Bühne hinzu.

Öffnen Sie nun die `CarouselMenu`-Klasse, und legen Sie das Klassengrundgerüst folgendermaßen an:

```
package{
    import flash.display.*;
    public class CarouselMenu extends MovieClip
    {
        public function CarouselMenu():void
        {
        }
    }
}
```

Um die `CarouselItem`-Klasse zu testen, werden Sie nun eine Instanz dieser Klasse erzeugen und der Bühne hinzufügen. Erweitern Sie den Konstruktor der `CarouselMenu`-Klasse zu diesem Zweck um folgende Codezeilen:

Schritt 7

Abgerundetes Rechteck zeichnen

Code in mehrere Klassen aufteilen

In größeren ActionScript-Projekten ist es sinnvoll, die einzelnen Funktionalitäten auf verschiedene Klassen aufzuteilen. So ist der Code übersichtlicher und Sie können Elemente ganz einfach ersetzen. In diesem Fall können Sie später die Karussellelemente bequem austauschen und zum Beispiel Bilder in jedes Element laden.

Schritt 8

Die CarouselMenu-Klasse

Schritt 9

Ein MenuItem erzeugen

Exkurs zur

3D-Programmierung

Eignen Sie sich die Grundlagen der 3D-Programmierung in Flash im Exkurs auf Seite 184 an.

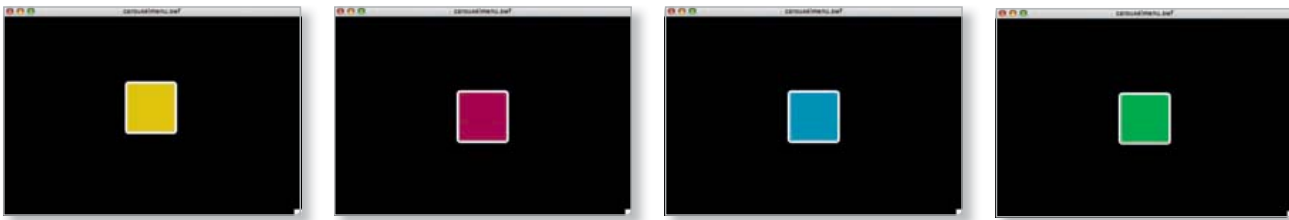
```
var item:CarouselItem = new CarouselItem();
item.x = stage.stageWidth/2;
item.y = stage.stageHeight/2;
addChild(item);
```

Nachdem Sie die Instanz erzeugt haben, positionieren Sie diese auf die Mitte der Bühne.

Schritt 10

Film testen

Testen Sie nun Ihren Flash-Film mit `Strg`/`⌘` + `↵`. Wie Sie sehen, erscheint ein farbiges Rechteck mit weißem Rahmen in der Mitte der Bühne. Wenn Sie den Film öfter testen, werden Sie feststellen, dass das Rechteck immer mit einer anderen, zufälligen Farbe gefüllt ist.



▲ Abbildung 4

Bei jedem neuen Test hat das dynamisch generierte Rechteck eine andere Farbe.

Schritt 11

3D-Positionsvariablen deklarieren

Damit Sie die `CarouselItem`-Instanzen im dreidimensionalen Raum verteilen und bewegen können, müssen Sie die Variablen für die x-, y- und z-Position im 3D-Raum deklarieren. Wechseln Sie also wieder in die `CarouselItem`-Klasse, und deklarieren Sie die Variablen wie folgt im Deklarationsbereich der Klasse über dem Konstruktor:

```
public var posZ:Number;
public var posX:Number;
public var posY:Number;
```

Durch das Attribut `public` haben Sie nun auch aus der `CarouselMenu`-Klasse Zugriff auf diese Klassenvariablen.

Schritt 12

3D-Umgebung vorbereiten

Wechseln Sie jetzt wieder in die `CarouselMenu`-Klasse. Hier müssen Sie jetzt im Deklarationsbereich die Variablen deklarieren, die zur Berechnung der 3D-Welt nötig sind.


```

1: private var focalLength:int = 250;
2: private var viewPointX:int = stage.stageWidth/2;
3: private var viewPointY:int = stage.stageHeight/2;
4: private var radius:uint = 400;
5: private var offsetZ:uint = 400;
6: private var numItems:uint = 10;
7: private var items:Array;

```

In den Zeilen 1 bis 3 legen Sie – wie schon aus den vorherigen Workshops bekannt – die Klassenvariablen für die Brennweite und den Aussichtspunkt fest. Des Weiteren deklarieren Sie in Zeile 4 die Variable `radius` und weisen dieser den Wert 400 zu. `radius` wird den Radius des Kreises festlegen, in dem die Menüelemente angeordnet werden. Zusätzlich deklarieren Sie in Zeile 5 eine Variable `offsetZ`, der Sie ebenfalls den Wert 400 zuweisen. `offsetZ` wird später alle Elemente im 3D-Raum um 400 Einheiten nach hinten in den Raum verschieben. Der Mittelpunkt des Kreises wäre ansonsten gleich dem Betrachtungspunkt, und die Karussellelemente würden so auch hinter der Kamera erzeugt werden.

In Zeile 6 legen Sie in der Variablen `numItems` fest, wie viele Elemente Sie in das Karussell aufnehmen. In diesem Falle sind es 10. Als Weiteres benötigen Sie ein Array `items`, in dem Sie die Karussellelemente speichern, um später die `sortZ`-Methode darauf anzuwenden. Dazu kommen wir aber erst später.

Um Elemente in einem Kreis zu positionieren, müssen Sie ein wenig Geometrie anwenden. Aber keine Angst, wie Sie sehen werden, ist das ganz einfach.

Der gesamte Quelltext innerhalb des Konstruktors der `CarouselMenu`-Klasse diene nur zu Testzwecken. Tauschen Sie den Code deswegen durch den folgenden aus:

```

1: items = new Array();
2: for (var i : uint = 0; i < numItems; i++)
3: {
4:     var item:CarouselItem = new CarouselItem();
5:     addChild(item);
6:     var angle : Number = (Math.PI * 2) / numItems * i;
7:     item.posZ = Math.sin (angle) * radius;
8:     item.posX = Math.cos (angle) * radius;
9:     item.posY = 100;
10:    items.push(item);
11: }

```

Schritt 13

Elemente im Karussell positionieren

Bevor Sie die Karussellelemente erzeugen, weisen Sie der Instanzvariablen `items` ein neues `Array` zu. In diesem werden die Karussellelemente gespeichert. In der `for`-Schleife ab Zeile 2 werden die Karussellelemente erzeugt. Die Schleife wird so oft durchlaufen, wie `numItems` groß ist. Der Klassenvariablen `numItems` hatten Sie den Wert 10 zugeordnet.

In Zeile 6 errechnen Sie `angle`, den Winkel (Radiant), mit dem das Karussellelement im Kreis platziert wird. Die Länge der Strecke um einen Kreis mit dem Radius 1 ist gleich $2 * \text{Pi}$. Wenn Sie diesen Wert durch die Gesamtanzahl der Kreiselemente teilen und das Ergebnis mit dem jeweiligen Index `i` multiplizieren, verteilen Sie die Winkel gleichmäßig auf alle Elemente.

In den Zeilen 7 und 8 legen Sie die z- und x-Position des Menüelements fest. Hierfür brauchen Sie nur der `cos`- bzw. `sin`-Methode `angle` als Parameter zu übergeben und diesen Wert `Math.sin` mit dem Radius `Math.cos` zu multiplizieren. Die y-Position eines jeden Elements setzen Sie auf 100.

Schritt 14

enterFrameListener hinzufügen

Da Sie zwar die `xPos`-, `yPos`- und `zPos`-Eigenschaften bestimmt, aber noch nicht die Formel programmiert haben, die diese Werte in die tatsächlichen x-, y- und Skalierungswerte im zweidimensionalen Koordinatensystem umrechnet, würde das Testen des Films jetzt wenig Sinn machen.

Zuerst benötigen Sie eine Methode, die in jedem Frame aufs Neue ausgeführt wird und diese Berechnung vornimmt. Zu dem Zweck müssen Sie das `flash.events`-Package in der `CarouselMenu`-Klasse unter der letzten Import-Anweisung im Deklarationsbereich importieren:

```
import flash.events.*;
```

Fügen Sie nun der Dokumentklasse `CarouselMenu` in der letzten Zeile des Konstruktors nach der `for`-Schleife einen `EventListener` hinzu, der in jedem Frame eine Methode `enterFrameListener()` aufruft:

```
this.addEventListener(Event.ENTER_FRAME,enterFrameListener);
```

Schritt 15

3D-Engine programmieren

Jetzt müssen Sie noch die `enterFrameListener`-Methode programmieren. Fügen Sie Ihrer Klasse die Methode folgendermaßen hinzu:

```
1: private function enterFrameListener(event:Event):void
2: {
```

```

3:
4:     for(var i:uint = 0; i < items.length; i++){
5:         var item:CarouselItem = CarouselItem(items[i]);
6:         //3D-Engine
7:         var scale:Number = focalLength/(focalLength + item.
            posZ + offsetZ);
8:         item.scaleX = item.scaleY = scale;
9:         item.x = viewPointX + item.posX * scale;
10:        item.y = viewPointY + item.posY * scale;
11:    }
12: }

```

Wie auch schon in den vorherigen 3D-Workshops wenden Sie nun die 3D-Formel aus dem 3D-Exkurs an. In Zeile 7 beziehen Sie außerdem noch die Klassenvariable `offsetZ` in die Berechnung von `scale` ein, damit das Karussell entsprechend dem Radius weiter nach hinten in den Raum versetzt wird. Andernfalls würde der Betrachtungspunkt innerhalb des Karussells sein.

Testen Sie nun Ihren Film. Das ist doch schon mal ein sehenswertes Ergebnis! Die zehn Rechtecke bilden einen Kreis, der in den Raum ragt.



Schritt 16

Film testen

◀ Abbildung 5

Flash positioniert die dynamisch erzeugten Rechtecke perspektivisch korrekt im Raum.

Wie Sie dem 3D-Exkurs und den vorherigen Workshops entnehmen können, müssen die Objekte im 3D-Raum eine Tiefe entsprechend ihrer z-Position erhalten. Fügen Sie dazu die Methode `sortZ()` ans Ende der `CarouselMenu`-Klasse hinzu:

Schritt 17

Z-Sortierung

```
private function sortZ():void
{
    items.sortOn("posZ", Array.DESENDING | Array.NUMERIC);
    for(var i:uint = 0; i < items.length; i++){
        var item:CarouselItem = items[i];
        setChildIndex(item,i);
    }
}
```

Die `sortZ`-Methode muss nun noch in jedem Frame aufgerufen werden. Schreiben Sie dazu den Methodenaufruf in die letzte Zeile der `enterFrameListener-Methode`: `sortZ();`

Schritt 18

Maussteuerung programmieren

In diesem Schritt geht es darum, das Karussell mit der Maus zu drehen. In der 3D-Programmierung können Sie eine Drehung um alle drei Achsen erzeugen. In diesem Fall werden Sie den Kreis um die y-Achse drehen. Die Formel für die y-Achsen-Rotation lautet:

$$x1 = \cos(\text{angleY}) * x - \sin(\text{angleY}) * z;$$

$$z1 = \cos(\text{angleY}) * z + \sin(\text{angleY}) * x;$$

In unserem Beispiel sind alle Werte bis auf `angleY` bekannt. Diesen Wert werden wir über die Mausposition berechnen. Ändern Sie die `enterFrameListener-Methode` folgendermaßen:

```
1: private function enterFrameListener(event:Event):void
2: {
3:     var angleY : Number = (mouseX - viewPointX) *.0002;
4:     var cosY : Number = Math.cos ( - angleY);
5:     var sinY : Number = Math.sin ( - angleY);
6:     for(var i:uint = 0; i< items.length; i++){
7:         var item:CarouselItem = CarouselItem(items[i]);
8:         var x1 : Number = item.posX * cosY - item.posZ * sinY;
9:         var z1 : Number = item.posZ * cosY + item.posX * sinY;
10:        item.posX = x1;
11:        item.posZ = z1;
12:        //3D Engine
13:        var scale:Number = focalLength/(focalLength + item.
posZ + offsetZ);
14:        item.scaleX = item.scaleY = scale;
15:        item.x = viewPointX + item.posX * scale;
16:        item.y = viewPointY + item.posY * scale;
```

```

17:    }
18:    sortZ();
19: }

```

In Zeile 3 errechnen Sie den `angleY`-Wert durch das Verhältnis der Mausposition zu `viewPointX`, der sich in der Mitte der Bühne befindet. Der Wert von `angleY` wird also, wenn sich die Maus rechts von dem Mittelpunkt befindet, größer Null sein, links davon kleiner Null.

Da `angleY` allerdings die `sin`- und `cos`-Methoden der `Math`-Klasse übergeben werden und diese mit Radiant rechnet, wäre das Ergebnis zu hoch und würde zu einer sehr schnellen Bewegung führen. Deswegen multiplizieren Sie das Ergebnis mit 0.0002. In den Zeilen 4 und 5 speichern Sie den Kosinus und Sinus von `angleY` in den Variablen `cosY` und `sinY`, damit die gleich angewandte Formel übersichtlicher zu lesen ist. Sie verwenden den negativen `angleY`-Wert, da die Drehung des Kreises immer entgegen der Mausrichtung stattfinden soll.

Die Rotationsformel wenden Sie in den Zeilen 8 und 9 an, um die neue `x`- und `z`-Position zu berechnen. In den Zeilen 10 und 11 weisen Sie der `posX`- und `posZ`-Eigenschaft die neuen Positionen zu. Die darauf folgende 3D-Engine erledigt wie gewohnt den Rest.

Testen Sie nun wieder Ihren Film. Bewegen Sie die Maus auf der Bühne. Wie Sie sehen, dreht sich der Kreis entgegen der Mausrichtung. Die Drehung wird schneller, je weiter Sie sich mit der Maus vom Mittelpunkt entfernen.



Schritt 19

Film testen

◀ **Abbildung 6**
Bereits jetzt können Sie das Menü mit der Maus bewegen.

Schritt 20

Blickwinkel ändern

Wie Sie beim Testen des Films gesehen haben, befindet sich der Kreis in der unteren Hälfte der Bühne. Das kommt daher, weil die Kreiselemente alle die y-Position 100 haben. Der Vorteil dieser y-Position ist, dass Sie so eine perspektivische Krümmung des Karussells erzeugen, da Sie von schräg oben darauf schauen. Um den Kreis mittig zu positionieren, können Sie ganz einfach den Blickwinkel über die `viewPointY`-Variable ändern. Weisen Sie dieser im Deklarationsbereich die Bühnenhöhe geteilt durch 3 (statt durch 2) zu. Ersetzen Sie daher die bisherige Variablendeklaration mit folgender Zeile:

```
private var viewPointY:int = stage.stageHeight/3;
```

Schritt 21

Film testen

Testen Sie nun Ihren Film. Wie Sie sehen, befindet sich der Kreis nun mittig auf der Bühne.

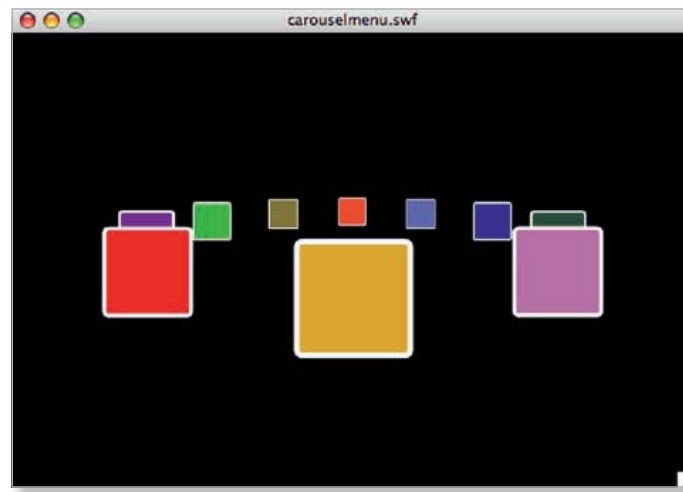


Abbildung 7 ►

Durch die Änderung des Aussichtspunktes befindet sich das Menü nun in der Mitte der Bühne – und das Karussell krümmt sich im Raum.

Schritt 22

Karussellelemente anklicken

Ziel ist es nun, den Karussellelementen einen Klick-Listener hinzuzufügen, um sie bei einem Mausklick jeweils an die vorderste Position des Karussells zu fahren. Um aus allen Methoden der Klasse `CarouselMenu` abfragen zu können, welches das angeklickte Element ist, ist es nötig, dieses in einer Klassenvariablen zu speichern.

Deklarieren Sie in der `CarouselMenu`-Klasse eine Variable `currentItem` vom Typ `CarouselItem`:

```
private var currentItem:CarouselItem;
```

Fügen Sie jetzt jedem `item` am Ende der `for`-Schleife im Konstruktor einen Klick-Listener hinzu, der beim Klick auf ein beliebiges Karussellelement die Methode `mouseDownListener()` aufruft:

```
item.addEventListener(MouseEvent.CLICK, mouseDownListener);
```

Programmieren Sie nun wie folgt die Methode `mouseDownListener()` am Ende der Klasse, um den Klick abzufangen:

```
private function mouseDownListener(event:MouseEvent):void
{
    currentItem = currentItem == CarouselItem(event.target) ?
        null : CarouselItem(event.target) ;
}
```

Bei dem ersten Klick auf ein Menüelement soll die Klassenvariable `currentItem` dieses Element referenzieren. Wenn wiederholt auf dasselbe Menüelement geklickt wird, soll `currentItem` wieder auf `null` gesetzt werden. Ziel des Ganzen ist, gleich in der `enterFrameListener`-Methode abzufragen, ob `currentItem` gesetzt ist, um dann das Karussell so zu drehen, dass das Element nach vorne fährt.

Ersetzen Sie die erste Zeile der `enterFrameListener`-Methode durch folgenden Code:

```
1: var angleY : Number;
2: if(currentItem != null)
3:     angleY = int(currentItem.posX - (currentItem.posX >= 0 ?
        (-radius-currentItem.posZ) : -(-radius-currentItem.posZ)))
        *.0002;
4: else
5:     angleY = (mouseX - viewPointX) *.0002;
```

In Zeile 1 deklarieren Sie die Methodenvariable `angleY`. Dann fragen Sie ab, ob `currentItem` ungleich `null` ist, also ob ein Menüelement angeklickt wurde. Um das Karussell nun so zu drehen, dass das angeklickte Element nach vorne fährt, muss `angleY` mit Hilfe des `posX`-Wert errechnet werden. Wenn `posX` gleich 0 ist, ist das Element vorne. Allerdings mit einer Einschränkung: Es gibt im Kreis zweimal einen Nullpunkt von `posX`. Einmal

Schritt 23

Karussell drehen

vorne und einmal hinten. Deswegen müssen Sie die Berechnung auch von der z-Position von `currentItem` abhängig machen. Je nachdem, ob `posX` größer oder kleiner als 0 ist, ziehen Sie von dieser die Differenz von `-radius` und `posZ` ab. Wenn sich das Element vorne befindet, ist die Differenz aus `-radius` und `posZ` gleich 0.

In den Zeilen 4 und 5 bestimmen Sie, was passiert, wenn `currentItem` nicht gesetzt ist. In diesem Fall wird `angleY` wieder von der Mausposition bestimmt.

Schritt 24

Film testen

Testen Sie nun wieder Ihren Film. Der Kreis dreht sich je nach der Mausposition zur Mitte der Bühne. Wenn Sie nun eines der Menüelemente anklicken, dreht sich der Kreis automatisch solange, bis das angeklickte Element ganz vorne ist. Klicken Sie jetzt auf ein anderes Element, wird dieser Vorgang entsprechend wiederholt. Wenn Sie auf ein bereits angeklicktes Element klicken, setzt die Maussteuerung wieder ein.

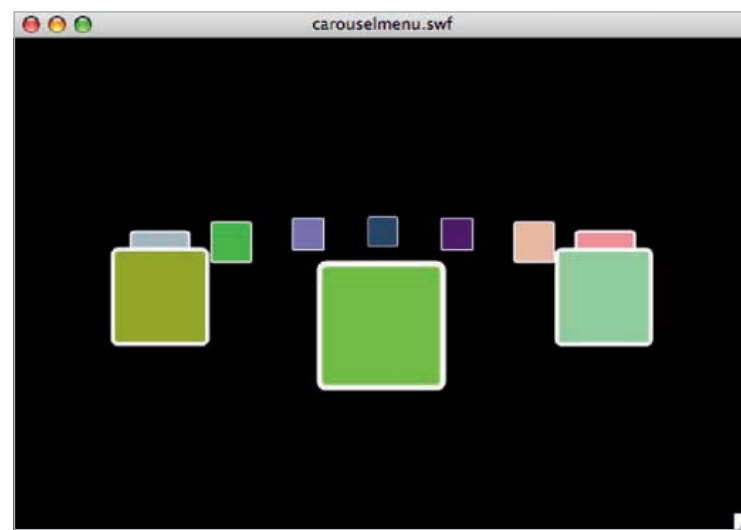


Abbildung 8 ►

Alle Menüelemente lassen sich durch einen Klick nach vorne holen.

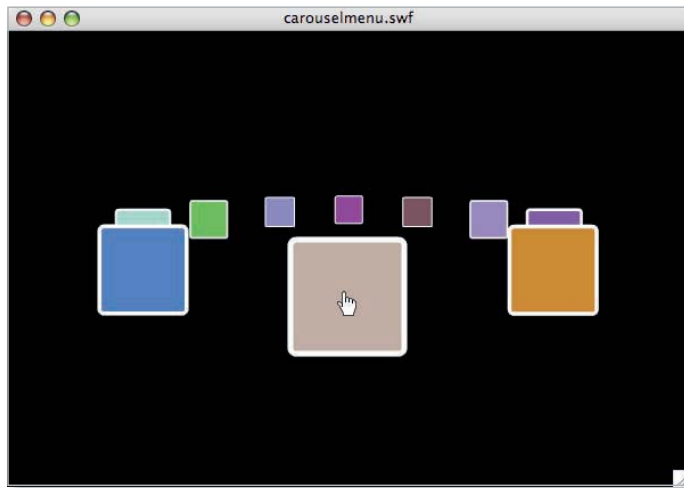
Schritt 25

Mauszeiger ändern

Damit die User Ihrer Site auch wissen, dass man die Menüpunkte anklicken kann, können Sie mit einer kleinen Zeile Code den Mauszeiger beim Überfahren der Elemente verändern. Sie müssen nichts weiter tun, als den `buttonMode` des Sprites auf `true` zu setzen. Fügen Sie dazu in die Carousel-Item-Klasse ans Ende des Konstruktors folgende Zeile ein:

```
this.buttonMode = true;
```


Wie Sie sehen, ändert sich der Mauszeiger in eine Klickhand, wenn Sie mit der Maus über ein Menüelement fahren.



Sie haben es geschafft! Sie haben ein Karussellmenü gebaut, das Sie mit Ihren neu erworbenen Kenntnissen leicht erweitern können. Sie können zum Beispiel eine Bildergalerie der besonderen Art erschaffen. Schauen Sie sich die Parallelen und Unterschiede zu dem vorherigen 3D-Workshop (3D-Bildergalerie) an. Mit ein paar Codezeilen können Sie die Bilder in Ihr 3D-Karussellmenü laden.

Schritt 26

Film testen

◀ Abbildung 9

Eine kleine Klickhand zeigt an, dass ein Element angeklickt werden kann.

Anwendungen und dynamische Inhalte

ClipLoader	234
Doppelklick-Klasse	245
Tooltips	253
Dateiupload	263
Individuelle Kontextmenüs	268
Laufband	279
Slideshow	290
XML	300
XML-Menü	302
Flash Remoting	309
Flash-Gästebuch	311

ClipLoader

Externe Bilder über eine eigene Loader-Klasse laden

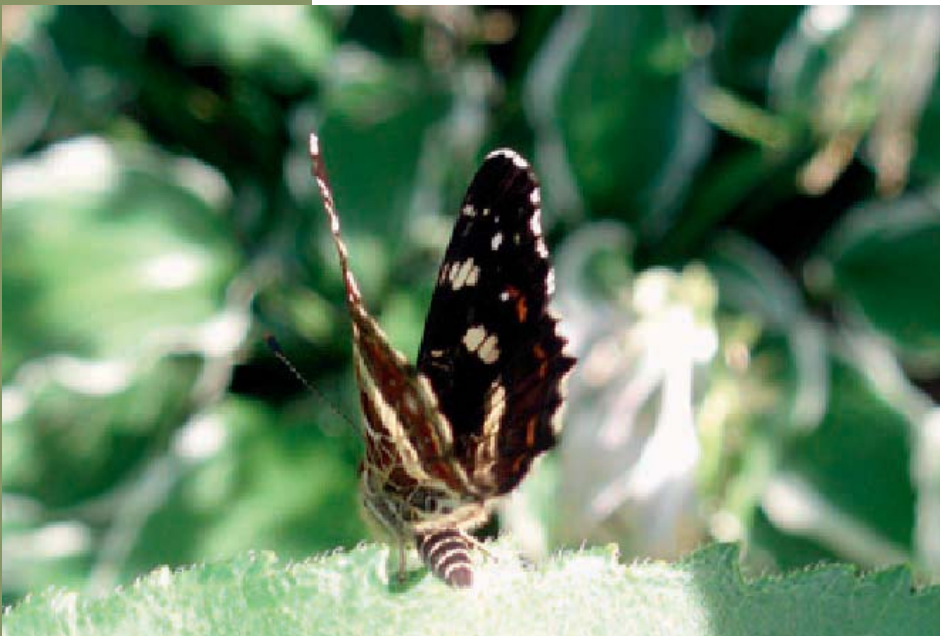
```
import mx.events.EventDispatcher;
class ClipLoader extends MovieClipLoader{
    private var dispatchEvent:Function;
    private var addEventListener:Function;
    private var removeEventListener:Function;
    private var initObjects:Object;
    public function ClipLoader()
    {
        super();
        EventDispatcher.initialize(this);
        this.initObjects = new Object();
    }
    public function loadClip(url:String, target:Object,
    initObj:Object):Void
    {
        this.initObjects[target] = initObj;
        super.loadClip(url, target);
    }
}
```

In diesem Workshop werden Sie eine wieder verwendbare Loader-Klasse programmieren, die Ihnen bei Ihren weiteren ActionScript 2-Projekten das Laden von externen Dateien erleichtern soll. Der Workshop soll Ihnen zeigen, wie Sie die Funktionalität der von Flash mitgelieferten Klassen erweitern können. Wer möchte, kann auch direkt zum letzten Schritt des Workshops springen, in dem mit der Loader-Klasse ein externes Bild geladen wird. Alle, die an der Programmierung der Klasse selbst interessiert sind, beginnen bei Schritt 1.

Zielsetzungen:

- ClipLoader-Klasse programmieren
- Ein externes Bild laden
- Ladebalken erzeugen
- Ladefortschritt anzeigen

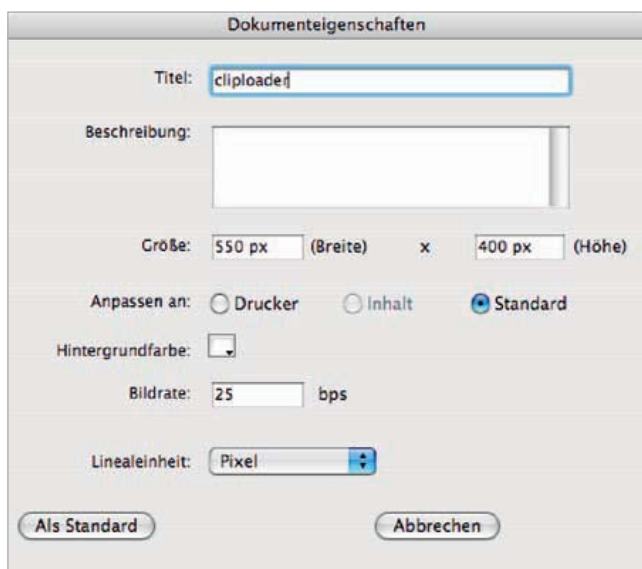
[Ergebnis: 21_CLIPLOADER •
CLIPLOADER.SWF]



Im ersten Schritt legen Sie einen neuen Ordner auf Ihrer Festplatte an und nennen ihn CLIPLOADER. Hier werden Sie die von Ihnen erstellten Beispiele zu diesem Workshop abspeichern.

Da Sie mit ActionScript 2 programmieren, beinhaltet die ActionScript-Datei sämtliche Funktionalität. Die fla-Datei dieses Workshops dient nur Testzwecken. Im praktischen Einsatz würden Sie dann die Flash-Datei Ihres jeweiligen Projektes nutzen.

Erstellen Sie ein neues AS2-Flash-Dokument mit der Grösse 550x400 Pixel und einer BILDRATE von 25 bps. HINTERGRUNDFARBE und sonstige Einstellungen spielen keine Rolle. Speichern Sie die Datei unter dem Namen CLIPLOADER.FLA in dem Projektordner.



Wählen Sie DATEI • NEU • ACTIONSCRIPT-DATEI. Speichern Sie die Datei unter dem Namen CLIPLOADER.AS in demselben Ordner wie die Flash-Datei.

Um externe Dateien wie zum Beispiel Bilder oder swf-Dateien dynamisch in Ihre Applikation zu laden, können Sie die Klasse `MovieClipLoader` oder die Methode `loadMovie()` der `MovieClip`-Klasse verwenden. Die Flash-Hilfe enthält sehr umfangreiche Informationen über beide Möglichkeiten.

Das Ziel dieses Workshops ist es, die `MovieClipLoader`-Klasse so zu erweitern, dass das Laden und insbesondere der Zugriff auf die Eigen-

Schritt 1 Vorbereitungen

Schritt 2 Flash-Dokument erstellen

◀ **Abbildung 1**
Das Flash-Dokument wird vorbereitet.

Schritt 3 Erstellen der ActionScript-Datei

Schritt 4 Ziel der ClipLoader-Klasse

Schritt 5 Erstellen der ClipLoader-Klasse

super() aufrufen

Würde die Klasse, von der Sie erben, Parameter erwarten, hätten Sie die Möglichkeit, über `super(Parameter)` dem Konstruktor der Super-Klasse diese Parameter zu übergeben, auch wenn Ihre eigene Klasse keine Konstruktorparameter erwarten sollte. Des Weiteren können Sie über `super.MethodenName()` eine Methode der Super-Klasse aufrufen.

attachMovie()

Auch in der `attachMovie`-Funktion kann man ein Initialisierungsobjekt übergeben, das die Eigenschaften des Movieclips bestimmt:

```
public attachMovie(id:  
String, name: String,  
depth: Number,  
[initObject:  
Object]) : MovieClip
```

Schritt 6 ClipLoader-Klasse testen

schaften der geladenen Datei vereinfacht wird. Sie werden die `MovieClipLoader`-Methode `loadClip()` um einen weiteren Parameter ergänzen: ein Initialisierungsobjekt, das die Eigenschaften des geladenen Movieclips bestimmt.

Wechseln Sie nun zu der Datei `CLIPLOADER.AS`, und schreiben Sie folgende Zeilen, um die Grundfunktionalität zu programmieren:

```
1: class ClipLoader extends MovieClipLoader{  
2:     public function ClipLoader()  
3:     {  
4:         super();  
5:     }  
6:     public function loadClip(url:String, target:Object,  
       initObj:Object):Void  
7:     {  
8:         super.loadClip(url, target);  
9:     }  
10: }
```

In Zeile 1, der Klassendefinition, bestimmen Sie, dass Ihre Klasse von `MovieClipLoader` erbt. Ihre Klasse besitzt somit die gesamte Funktionalität von `MovieClipLoader`. Im Konstruktor, Zeile 2 bis 4, rufen Sie mit `super()` den Konstruktor von `MovieClipLoader` auf.

Ab Zeile 6 überschreiben Sie die `loadClip`-Methode von `MovieClipLoader`. In Zeile 8 rufen Sie dann die `loadClip`-Methode von `MovieClipLoader` auf und übergeben ihr die notwendigen Parameter. Wenn Sie die `loadClip`-Methode von `MovieClipLoader` mit Ihrer `ClipLoader`-Klasse vergleichen, fällt auf, dass Ihre Methode einen Parameter mehr erwartet: `initObj` vom Typ `Object`. Dieses Objekt wird dazu dienen, bereits beim Aufruf von `loadClip()` ein Initialisierungsobjekt zu übergeben, mit dem Sie die Eigenschaften des zu ladenden Movieclips schon vor dem Ladevorgang bestimmen können.

Vergleichen Sie dazu in der Flash-Hilfe die Methode `attachMovie()`, die dasselbe Prinzip nutzt.

Wechseln Sie nun in Ihre Flash-Datei. Klicken Sie auf das erste Bild der ersten Ebene in der Hauptzeitleise, und öffnen Sie mit `[F9]` oder `[Alt] + [F9]` am Mac das `ActionScript`-Fenster. Geben Sie folgenden Code ein:

```

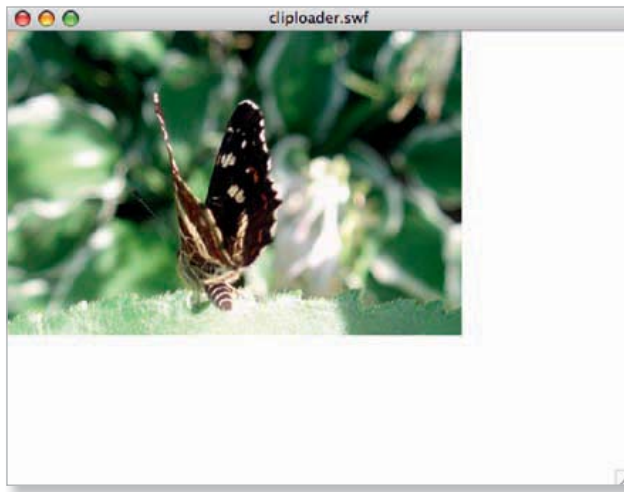
1: var cl:ClipLoader = new ClipLoader();
2: var mc:MovieClip = this.createEmptyMovieClip('mc',1);
3: cl.loadClip('http://www.flashfastforward.de/images/image1.
   jpg',mc);

```

In Zeile 1 erzeugen Sie eine Instanz der ClipLoader-Klasse und speichern diese in der Variable `cl`. Als Nächstes erstellen Sie mit `createEmptyMovieClip()` einen Movieclip, in den die externe Datei geladen werden soll. In Zeile 3 rufen Sie nun die `loadClip`-Methode von `cl` auf und übergeben dieser eine URL zu einer Bilddatei und den `mc` als `target`.

Testen Sie nun den Flash-Film mit `[Strg]/[⌘]+[↵]`.

Nach einer kurzen Ladezeit erscheint das geladene Bild auf der Bühne. Ihre ClipLoader-Klasse funktioniert!



Damit das Initialisierungsobjekt als dritter Parameter der `loadClip`-Methode nun auch seine Wirkung zeigen kann, und die Eigenschaften der geladenen Datei bestimmt, erweitern Sie die ClipLoader-Klasse um die Zeilen 2 und 6:

```

1: class ClipLoader extends MovieClipLoader {
2:     private var initObjects:Object;
3:     public function ClipLoader()
4:     {
5:         super();
6:         this.initObjects = new Object();
7:     }
8:     ...

```

Pfadangabe zur Bilddatei

In diesem Fall befindet sich das Bild auf einem externen Server. Sie könnten zum Beispiel auch einen Pfad zu einem Bild im Projektordner angeben.

Schritt 7

Film testen

◀ Abbildung 2

Das Bild von der Buch-Website www.flashfastforward.de wird dynamisch in den Flash-Film geladen.

Schritt 8

Initialisierungsobjekt hinzufügen

In Zeile 2 deklarieren Sie eine Variable `initObjects`, der Sie dann im Konstruktor in Zeile 6 ein neues Objekt zuweisen. Dieses Objekt wird als Datenbehälter dienen, um sämtliche Initialisierungsobjekte, die der `loadClip`-Methode übergeben werden, zu speichern.

Schritt 9

Initialisierungsobjekt speichern

Erweitern Sie jetzt die `loadClip`-Methode um die Zeile 3:

```
1: public function loadClip(url:String, target:Object,  
    initObj:Object):Void  
2: {  
3:     this.initObjects[target] = initObj;  
4:     super.loadClip(url, target);  
5: }
```

In Zeile 3 erzeugen Sie in `initObjects` eine neue Objekteigenschaft. Als Bezeichner dieser Eigenschaft dient das übergebene `target`. Als Wert weisen Sie der Eigenschaft das Initialisierungsobjekt zu, das später die Eigenschaften der geladenen Datei `target` bestimmen soll. Dieses Objekt können Sie später dann ganz einfach über `initObjects[target]` abrufen.

Schritt 10

Initialisierung der Datei vornehmen

Die `MovieClipLoader`-Klasse besitzt die Methode `onLoadInit()`, die aufgerufen wird, wenn die Datei so weit geladen ist, dass man auf deren Eigenschaften zugreifen kann. Da Ihre Klasse von `MovieClipLoader` erbt, werden Sie diese Methode nutzen, um das im vorherigen Schritt gespeicherte Initialisierungsobjekt auf den geladenen Movieclip anwenden zu können. Übernehmen Sie dazu folgenden Code unterhalb der `loadClip`-Methode:

```
private function onLoadInit(target:MovieClip):Void{  
    for(var i in initObjects[target]){  
        target[i] = initObjects[target][i];  
    }  
    delete(initObjects[target]);  
}
```

In der zweiten Zeile durchlaufen Sie das gespeicherte Initialisierungsobjekt in einer `for-in`-Schleife und weisen die darin befindlichen Attribute und deren Werte denen von `target` zu. Nach getaner Arbeit ist das Initialisierungsobjekt nicht mehr brauchbar, und Sie löschen es mit `delete()`.

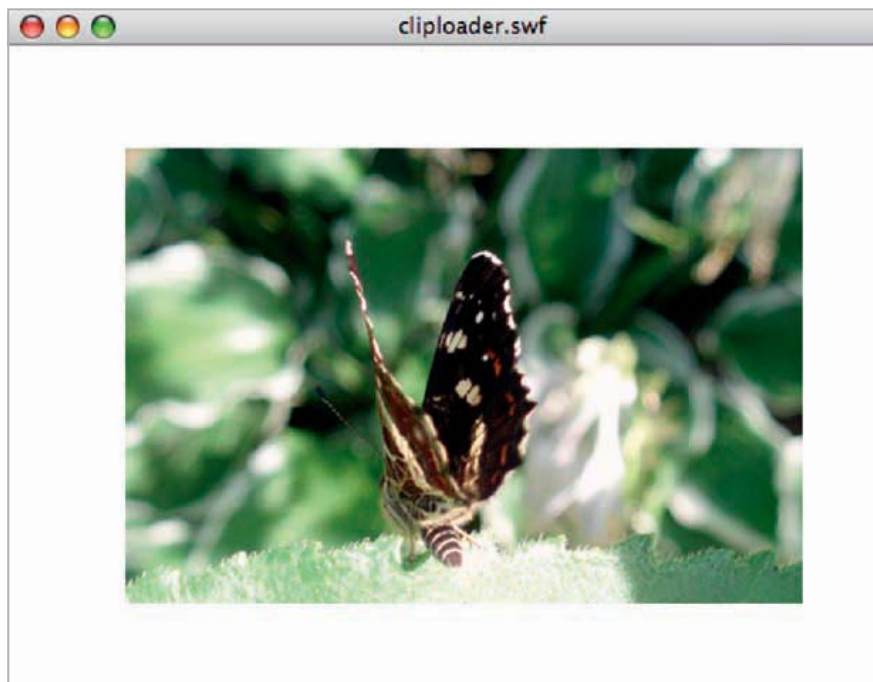
Wechseln Sie wieder zum Skript in der Hauptzeitleiste der Flash-Datei. Fügen Sie dem `loadClip`-Aufruf in Zeile 3 als weiteren Parameter das Initialisierungsobjekt `{_x:70,_y:60}` hinzu:

```
cl.loadClip('http://www.flashfastforward.de/images/image1.jpg',mc,{_x:70,_y:60});
```

Sie bestimmen damit, dass `mc` nach dem Ladevorgang die x-Position 70 und die y-Position 60 einnehmen soll.

Testen Sie jetzt wieder den Flash-Film mit `Strg`/`⌘` + `↵`.

Nach einer kurzen Ladezeit erscheint das geladene Bild und nimmt tatsächlich die von Ihnen definierten Eigenschaften an.



▲ Abbildung 3

Über das Initialisierungsobjekt können Sie x- und y-Position des geladenen Bildes bestimmen.

Wenn das Laden einer Datei scheitern sollte, wäre dessen Initialisierungsobjekt weiterhin in `initObjects` gespeichert und würde so unnötig Speicher verbrauchen.

Schritt 11

Testen des Initialisierungsobjekts

Schritt 12

Film testen

Methoden überschreiben

Wenn eine Klasse von einer anderen erbt, stehen der beerbten Klasse die Methoden der sogenannten Super-Klasse automatisch zur Verfügung. Wenn in der beerbten Klasse eine Methode deklariert wird, die auch in der Super-Klasse deklariert ist, ist damit diese Methode der Super-Klasse überschrieben. Bei dem Aufruf der Methode wird nur noch der Code aus der beerbten Klasse ausgeführt. Durch den Aufruf von `super.MethodName()` kann die Methode der Super-Klasse jedoch weiterhin aus der beerbten Klasse heraus aufgerufen werden.

Schritt 13

Ladefehler berücksichtigen

Fehlerursachen

Ein Fehler beim Laden einer Datei kann aus verschiedenen Gründen vorkommen. Die häufigste Ursache ist, dass die Datei nicht vorhanden ist oder dass keine Leserechte gegeben sind.

Um das zu verhindern, überschreiben Sie nun die `onLoadError`-Methode der `MovieClipLoader`-Klasse, die im Falle eines Ladefehlers aufgerufen wird. Wechseln Sie also wieder in die `ClipLoader`-Klasse, und übernehmen Sie unter der letzten Methode folgenden Code:

```
1: private function onLoadError(target:MovieClip,
   errorCode:String, httpStatus:Number):Void
2: {
3:     trace("ClipLoader: onLoadError");
4:     trace("target: "+target);
5:     trace("errorCode: "+errorCode);
6:     trace("httpStatus: "+httpStatus);
7:     delete(initObjects[target]);
8: }
```

Von Zeile 3 bis Zeile 6 geben Sie mit `trace()` die Informationen und Fehlermeldungen aus, die zum Abbruch des Ladevorgangs geführt haben. So können Sie beim Testen Ihres Projekts sofort im Ausgabefenster von Flash sehen, ob bei dem Ladevorgang etwas schiefgelaufen ist. In Zeile 7 löschen Sie das dann überflüssige Initialisierungsobjekt.

Schritt 14 Ladestatus anzeigen

Im Grunde haben Sie bereits ganze Arbeit geleistet und eine voll funktionsfähige, wieder verwendbare Loader-Klasse programmiert. Was ist allerdings mit der Wartezeit, bis die externe Datei geladen wurde? In dieser Zeit passiert zumindest für den Betrachter scheinbar gar nichts. Das wollen wir ändern. Im weiteren Verlauf des Workshops werden Sie mit Hilfe der `EventDispatcher`-Klasse ein Event senden, das den Prozentwert des Ladestatus übermittelt. Das gibt Ihnen die Möglichkeit, den Ladestatus anzuzeigen.

Schritt 15 Implementieren des EventDispatchers

Um ein eigenes Event zu senden, können Sie die `EventDispatcher`-Klasse verwenden.

Wechseln Sie also wieder zur `ClipLoader`-Klasse, und importieren Sie in der ersten Zeile die folgende Klasse:

```
import mx.events.EventDispatcher;
```

Um die Methoden der `EventDispatcher`-Klasse verwenden zu können, fügen Sie unterhalb der in Schritt 8 eingeführten Variable `initObjects` folgenden Code ein:

```
private var dispatchEvent:Function;  
private var addEventListener:Function;  
private var removeEventListener:Function;
```

Danach müssen Sie den `EventDispatcher` initialisieren, um dessen Funktionalität nutzen zu können. Übernehmen Sie dazu die folgende Zeile im Konstruktor direkt unter den Aufruf von `super()`:

```
EventDispatcher.initialize(this);
```

Um ein Event zu senden, verwenden Sie die Methode `dispatchEvent()`. Diese überträgt ein sogenanntes Event-Objekt, dessen Eigenschaften Sie selbst bestimmen können. Ziel ist es, darin den Prozentwert des Ladezustats zu übertragen. Den Ladestatus selbst errechnen Sie mit der `MovieClipLoader`-Methode `onLoadProgress()`, die automatisch und wiederholt während des Ladevorgangs aufgerufen wird. Übernehmen Sie dazu folgende Codezeilen am Ende Ihrer Klasse:

```
1: private function onLoadProgress(target:MovieClip,loadedBytes:  
   Number,totalBytes:Number):Void  
2: {  
3:     var percent:Number = Math.round(loadedBytes/  
       totalBytes*100);  
4:     this.dispatchEvent({type:'onPercentLoaded',target:target,  
       value:percent});  
5: }
```

In Zeile 3 berechnen Sie den Prozentwert des Ladezustats und speichern diesen in einer Variable `percent`. In Zeile 4 senden Sie nun das Event mit einem Event-Objekt. `type` legt dabei den Typ des Events fest. Den Typ nennen Sie `onPercentLoaded`. Mit der Event-Objekteigenschaft `target` übertragen Sie eine Referenz auf den zu ladenden Movieclip. Die Eigenschaft `value` übermittelt den Wert von `percent`. Ein Listener kann nun auf dieses Event lauschen und die Werte auslesen.

Schritt 16

Erzeugen eines Events

Schritt 17

Listener für das Event registrieren

Da die ClipLoader-Klasse von `MovieClipLoader` erbt, besitzt sie die Methode `addListener()`, über die bereits Listener hinzugefügt werden können. Jedoch heißt die Methode, mit der man mit Hilfe des EventDispatchers einen Listener hinzufügt, `addEventListener()`. Um die Listener, die sich bei der ClipLoader-Klasse registrieren, auch bei dem eigenen Event zu registrieren, müssen Sie die Methode `addListener()` überschreiben und dort auch `addEventListener()` aufrufen. Fügen Sie unter die eben eingefügte Methode folgenden Code ein und überschreiben Sie damit die `addListener`-Methode:

```
1: public function addListener(obj:Object):Void
2: {
3:     super.addListener(obj);
4:     this.addEventListener('onPercentLoaded',obj);
5: }
```

In Zeile 3 rufen Sie die `addListener`-Methode der `MovieClipLoader`-Klasse auf und übergeben dieser das Listener-Objekt `obj`. In der nächsten Zeile fügen Sie Ihrer Klasse mit Hilfe des EventDispatchers dasselbe Listener-Objekt hinzu.

Dasselbe müssen Sie nun auch mit der `removeListener`-Methode von `MovieClipLoader` machen, die dazu dient, die Listener wieder zu entfernen. Die entsprechende Methode der EventDispatcher-Klasse heißt `removeEventListener()`. Überschreiben Sie zu diesem Zweck auch diese Methode und ergänzen Sie direkt unter der `addListener`-Methode folgende Codezeilen:

```
public function removeListener(obj:Object):Void
{
    super.removeListener(obj);
    this.removeEventListener('onPercentLoaded',obj);
}
```

Schritt 18

Ladebalken erzeugen

Wechseln Sie nun wieder zum Skript in der Hauptzeitleiste Ihres Flash-Dokuments. Ziel ist es, einen einfachen Ladebalken zu erzeugen. Dazu verwenden Sie die `BitmapData`-Klasse in Verbindung mit der `Rectangle`-Klasse. Importieren Sie die beiden Klassen folgendermaßen am Anfang des Skripts:

```
import flash.geom.Rectangle;
import flash.BitmapData;
```

Erzeugen Sie nun am Ende des Skripts eine neue Instanz von `BitmapData` sowie einen neuen `Movieclip loader_mc`, dem Sie die `BitmapData`-Instanz per `attachBitmap` hinzufügen:

```
var bmd:BitmapData = new BitmapData(100, 8, false, 0x999999);
var loader_mc:MovieClip = this.createEmptyMovieClip(
    "loader_mc", 2);
loader_mc.attachBitmap(bmd, 1);
```

Positionieren Sie nun den Ladebalken direkt unter die letzten Codezeilen in der Mitte der Bühne:

```
loader_mc._x = Stage.width/2-loader_mc._width/2;
loader_mc._y = Stage.height/2-loader_mc._height/2;
```

Testen Sie nun Ihren Flash-Film mit `[Strg]/[⌘]+[↵]`. Wie Sie sehen, befindet sich ein grauer Balken auf der Bühne. Dieser zeigt allerdings noch nicht den Ladefortschritt an und wird außerdem auch über dem geladenen Bild angezeigt. All das erledigen Sie im nächsten Schritt.

Als Listener der `ClipLoader`-Instanz soll die Hauptzeitleiste, also `_root`, dienen.

Schreiben Sie nun in Ihre Flash-Datei die Funktion `onPercentLoaded`, und fügen Sie dem `ClipLoader cl` die `_root`-Zeitleiste als Listener hinzu. Ergänzen Sie ans Ende Ihres Skripts folgenden Code:

```
1: function onPercentLoaded (event:Object){
2:     bmd.fillRect(new Rectangle(0, 0, event.value, 8),
3:         0x000000);
4:     if(event.value == 100){
5:         loader_mc.removeMovieClip();
6:     }
7:     cl.addListener(this);
```

In Zeile 2 zeichnen Sie in das `BitmapData`-Objekt `bmd` ein Rechteck, dem als Breite die `value`-Eigenschaft des Event-Objekts `event` zugewiesen wird. In Zeile 3 prüfen Sie, ob der Prozentwert 100 erreicht hat und demnach

Schritt 19

Film testen

Schritt 20

Ladefortschritt anzeigen



▲ **Abbildung 4**
Der Ladebalken wird angezeigt, ist allerdings noch nicht funktionsfähig.

die Datei vollständig geladen wurde. In diesem Fall löschen Sie `loader_mc` und somit den Ladebalken von der Bühne.

Schritt 21

Film testen

Testen Sie nun Ihren Flash-Film mit `Strg`/`⌘` + `↵`. Wie Sie sehen, wächst in dem grauen Balken entsprechend des Ladefortschritts ein schwarzer Balken. Je nach Internetgeschwindigkeit kann dieser Vorgang auch schnell abgeschlossen sein. Hat der Ladebalken das Ende erreicht, ist der Ladevorgang beendet und der Ladebalken wird gelöscht. Sie haben ganze Arbeit geleistet.

Sie können die `ClipLoader`-Klasse nun ganz einfach in Ihren Projekten einsetzen. Denn bedenken Sie: Es sind im Grunde nur noch zwei Zeilen nötig, um eine externe Datei zu laden!

```
1: var cl:ClipLoader = new ClipLoader();
2: cl.loadClip('http://www.flashfastforward.de/images/image1.jpg',target);
```

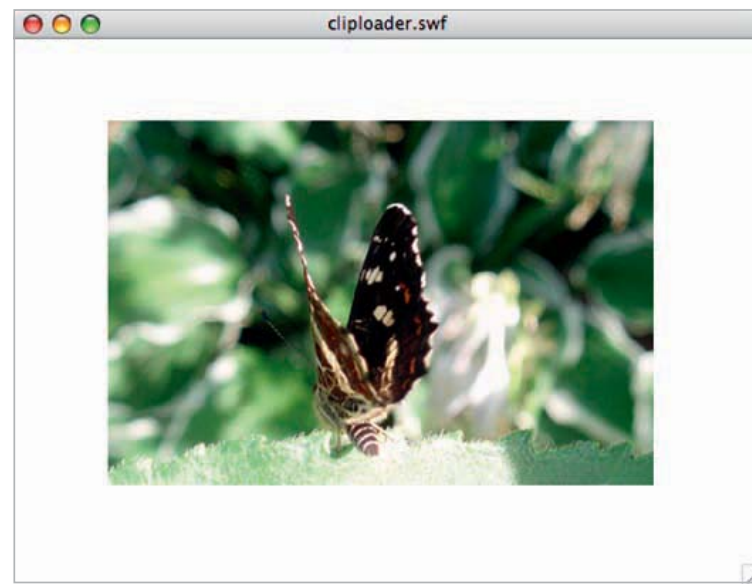


Abbildung 5 ►

Der Ladebalken verschwindet von der Bühne, sobald das Bild vollständig geladen ist und angezeigt werden kann.

Doppelklick-Klasse

Bilder per Doppelklick ein- und auszoomen

*Da es bis Flash CS3 und
ActionScript 3 keine
Doppelklick-Funktionali-
tät in Flash gab, werden
Sie in diesem Workshop
lernen, einen Doppel-
klick mit ActionScript 2
abzufangen und in Ihren
AS2-Flash-Projekten
einzusetzen.*

Zielsetzungen:

Bild importieren

Doppelklick
programmieren

Bild bei Doppelklick
vergrößern und wieder
verkleinern

[Ergebnis: 22_DOPPEL-
KLICK-KLASSE • DOUBLE-
CLICK.SWF]

```
class DoubleClick
{
    private static var clickObj:Object = new Object();
    private static var clickSpeed:Number = 200;
    public static function isDoubleClick(target:Object,
/*optional*/ speed:Number):Boolean
    {
        speed = speed || clickSpeed;
        if(clickObj[target] == undefined)
        {
            clickObj[target] = new Object();
            clickObj[target].clickinterval =
                setInterval(deleteClickObject, speed, target);
        }
        else
        {
```



Schritt 1 Vorbereitungen

Im ersten Schritt legen Sie einen neuen Ordner auf Ihrer Festplatte an und nennen diesen **DOPPELCLICK**. Hier werden Sie die von Ihnen erstellten Beispiele zu diesem Workshop abspeichern.

Schritt 2 Dokumente erstellen

Erstellen Sie ein neues AS2-Flash-Dokument mit der Grösse 550x400 Pixel und einer BILDRATE von 25bps. HINTERGRUNDFARBE und sonstige Einstellungen spielen keine Rolle. Speichern Sie die Datei unter dem Namen **DOUBLECLICK.FLA** im Projektordner.

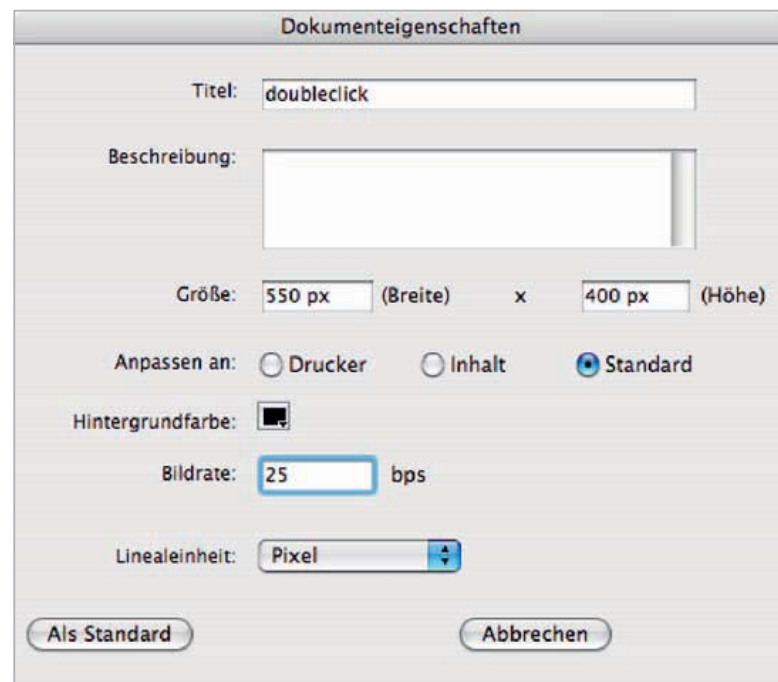


Abbildung 1 ►
Legen Sie über **[Strg]/**
[⌘] + [J] die Dokument-
eigenschaften fest.

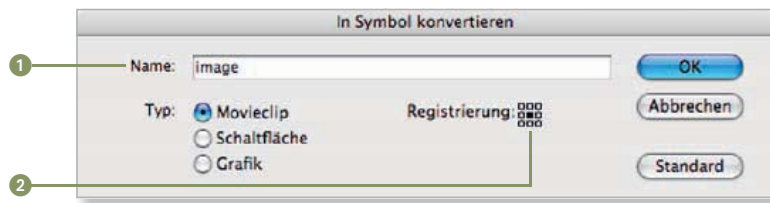
Schritt 3 ActionScript-Datei erzeugen

Wählen Sie **DATEI • NEU • ACTIONSCRIPT-DATEI**. Speichern Sie die Datei unter dem Namen **DOUBLECLICK.AS** im Projektordner.

Schritt 4 Flash-Datei vorbereiten

Importieren Sie über **DATEI • IMPORTIEREN • IN BÜHNE IMPORTIEREN** die Datei **IMAGE1.JPG**. Wenn Sie ein eigenes Bild nutzen wollen, sollte es eine maximale Größe von 550 x 400 Pixel haben.

Das Bild sollte sich nun zentriert auf der Bühne befinden. Klicken Sie jetzt auf das Bild, und konvertieren Sie es mit **[F8]** in ein Symbol vom Typ **MOVIECLIP**. Geben Sie diesem den Namen »image« **①**, und setzen Sie die **REGISTRIERUNG** auf die **Mitte** **②**. Bestätigen Sie Ihre Eingabe.



Geben Sie dem neu erzeugten Movieclip auf der Bühne im Eigenschaftsfenster den Instanznamen »image« (3).

Später soll über einen Doppelklick auf das Bild ein Zoomeffekt erzeugt werden.

Um die `DoubleClick`-Klasse global in einem Projekt zu verwenden, wird diese Klasse statische Methoden bereitstellen. Statische Methoden haben den Vorteil, einfach über `KlassenName.methodName()` aufgerufen werden zu können. Es muss also zuvor kein Objekt der Klasse erzeugt werden.

Öffnen Sie nun die Datei `DOUBLECLICK.AS`. Fügen Sie folgenden Quelltext ein:

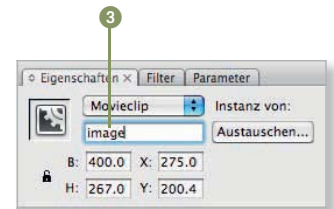
```
1: class DoubleClick
2: {
3:     private static var clickObj:Object = new Object();
4:     private static var clickSpeed:Number = 200;
5: }
```

Die Variable `clickObj` des Typs `Object` wird als Datenspeicher dienen. In ihr wird später eine Referenz auf das Objekt (Movieclip, Textfeld usw.) gespeichert, das auf einen Doppelklick hin geprüft werden soll. Die Variable `clickSpeed` legt die Zeit in Millisekunden fest, in der ein weiterer Klick erfolgt sein muss, damit es sich um einen Doppelklick handelt.

Die Methode `isDoubleClick()`, in der Sie den Doppelklick abfragen, funktioniert nach einem einfachen und doch wirkungsvollen Prinzip: Für jeden Klick, zum Beispiel auf einen Movieclip, wird ein Objekt mit einer Referenz auf diesen Movieclip erzeugt. Diese Referenz wird nach einer bestimmten Zeit wieder gelöscht. Sollte während dieser Zeit ein weiterer Klick auf den Movieclip erfolgen, stellt die Methode fest, dass bereits ein Objekt mit der entsprechenden Referenz auf diesen Movieclip vorhanden ist. Somit

◀ Abbildung 2

Das Bild wird in einen Movieclip umgewandelt.



▲ Abbildung 3

Vergeben Sie einen eindeutigen Instanznamen.

Schritt 5

DoubleClick-Klasse anlegen

Schritt 6

Variablen deklarieren

Geschwindigkeit eines Doppelklicks

Sie kennen dies zum Beispiel von Ihrem Betriebssystem. Auch da muss man innerhalb einer kurzen Zeit den zweiten Klick ausgelöst haben, damit eine Aktion ausgeführt wird.

Schritt 7

Auf einen Doppelklick prüfen

»weiß« die Methode, dass ein weiterer Klick, also ein Doppelklick, erfolgt ist. Schreiben Sie die Methode folgendermaßen unter die eben deklarierten Variablen:

```
1: public static function isDoubleClick(target:Object,  
    speed:Number):Boolean  
2: {  
3:     speed = speed || clickSpeed;  
4:     if(clickObj[target] == undefined)  
5:     {  
6:         clickObj[target] = new Object();  
7:         clickObj[target].clickinterval =  
            setInterval(deleteClickObject, speed, target);  
8:     }  
9:     else  
10:    {  
11:        deleteClickObject(target);  
12:        return true;  
13:    }  
14:    return false;  
15: }
```

Die Methode `isDoubleClick()` ist vom Typ `Boolean`, muss also `true` oder `false` zurückgeben. Sie erwartet zwei Parameter:

- ▶ `target` ist zum Beispiel ein Movieclip oder ein Button, der auf einen Doppelklick reagieren soll.
- ▶ `speed` setzt optional die Dauer des Intervalls, in dem der zweite Klick stattfinden muss.

Ist dieser Parameter nicht gesetzt und damit `undefined`, wird `speed` in Zeile 3 der im Deklarationsbereich festgelegte `clickSpeed`-Wert von 200 Millisekunden zugeordnet. In Zeile 4 wird nun geprüft, ob `clickObj` **kein** Objekt mit der Referenz auf das übergebene `target` enthält. Ist dies der Fall, erzeugen Sie dieses Objekt in Zeile 6.

In der nächsten Zeile starten Sie mit `setInterval()` ein Intervall, das eine Methode `deleteClickObject()` nach der Zeit `speed` aufruft und `target` als Parameter übergibt. Die Methode `deleteClickObject()`, die Sie im nächsten Schritt programmieren werden, wird das erzeugte Objekt und das Intervall wieder löschen. Für den Fall, dass bereits ein Objekt mit der Referenz auf `target` vorhanden ist und also bereits einmal auf `target` geklickt wurde, wird ab Zeile 9 der `else`-Rumpf ausgeführt.

Da Sie hier wissen, dass ein Doppelklick ausgeführt wurde, löschen Sie das erzeugte Objekt und das Intervall wieder mit dem Aufruf von `deleteClickObj(target)`. Der Doppelklick ist erfolgt, also geben Sie `true` zurück.

Sollte der `else`-Rumpf nicht ausgeführt worden sein, ist kein zweiter Klick erfolgt und Sie geben in Zeile 14 `false` zurück. Nehmen Sie sich ruhig die Zeit, die Funktionalität dieser Methode zu verinnerlichen.

Wie erwähnt fehlt noch die Methode `deleteClickObject()`, die von `isDoubleClick()` durch ein Intervall aufgerufen wird. Sie dient dazu, das Objekt mit der Referenz auf `target` sowie das Intervall wieder zu löschen.

Da das Löschen nur der `DoubleClick`-Klasse erlaubt sein soll, deklarieren Sie die Methode als `private`. Fügen Sie unter die eben erstellte Methode `isDoubleClick()` folgende Codezeilen ein:

```
1: private static function deleteClickObject(target:Object):Void
2: {
3:     clearInterval(clickObj[target].clickinterval);
4:     delete(clickObj[target]);
5: }
```

In Zeile 3 löschen Sie das Intervall, das diese Methode aufgerufen hat. In der nächsten Zeile löschen Sie das zu `target` gehörende Objekt aus dem `clickObj`-Speicher.

Wechseln Sie nun wieder zu der Flash-Datei, und erzeugen Sie eine weitere Ebene auf der Hauptzeitleiste. Klicken Sie in das erste Bild der neuen Ebene, und öffnen Sie das ActionScript-Fenster mit `[F9]` bzw. `[Alt] + [F9]` am Mac. Schreiben Sie folgende Zeilen, um den Doppelklick zu testen:

```
image.onPress = function(){
    var isDoubleClick:Boolean = DoubleClick.isDoubleClick(this);
    trace(isDoubleClick);
}
```

Testen Sie nun mit `[Strg]/[⌘] + [↵]` Ihren Flash-Film, und doppelklicken Sie auf das Bild. Wie Sie sehen, wird – je nachdem, wie schnell Sie geklickt haben – beim ersten Klicken `false` und beim zweiten Klicken `true` ausgegeben. Klicken Sie ruhig auch einmal in verschiedenen Abständen.

Schritt 8

deleteClickObject-Methode erstellen

Private und public

Damit Methoden oder Variablen einer Klasse von außen ansprechbar sind, deklariert man sie als `public`. Methoden oder Variablen, auf die nur die Klasse bzw. das Objekt der Klasse selbst Zugriff haben soll, deklariert man als `private`.

Schritt 9

Der Doppelklick im Einsatz

Schritt 10

Film testen

Herzlichen Glückwunsch, Sie haben Flash um die Doppelklick-Funktionalität erweitert und können nun in Ihren weiteren Flash-Projekten mit nur einer Zeile Code einen Doppelklick abfangen.

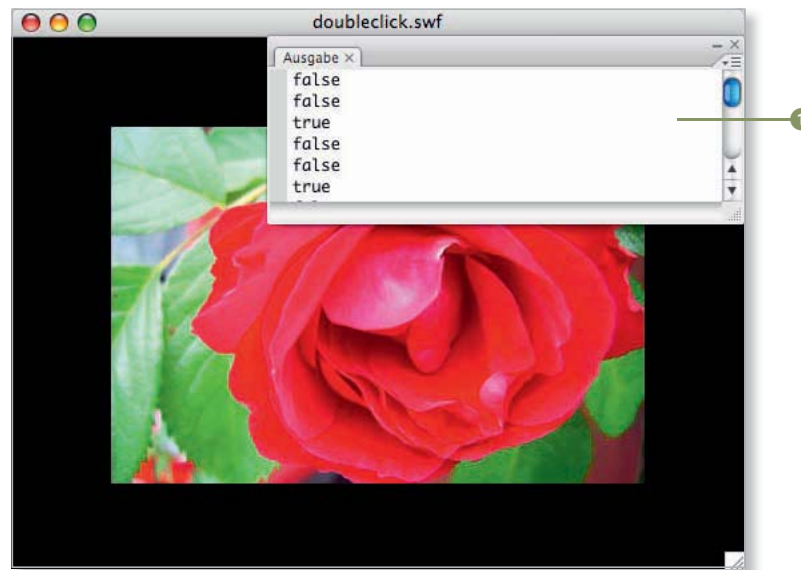


Abbildung 4 ►

Im Ausgabefenster ❶ können Sie mitverfolgen, wie die Klasse arbeitet.

Schritt 11 **Einen Zoomeffekt erstellen**

Da wir es nicht bei diesem Erfolg belassen wollen, werden Sie jetzt noch ein kleines Anwendungsbeispiel programmieren. Wie wäre es damit, das Bild bei einem Doppelklick zu vergrößern und zu verkleinern? Für die Animation können Sie die Tween-Klasse verwenden.

Importieren Sie diese und die Bewegungsklasse `Back` aus dem `mx.transitions.easing`-Package am Anfang Ihres Skripts auf der Hauptzeitleiste:

```
import mx.transitions.Tween;  
import mx.transitions.easing.Back;
```

Um das Bild bei einem darauf ausgeführten Doppelklick sowohl zu vergrößern als auch zu verkleinern, werden Sie die `_xscale`- und die `_yscale`-Eigenschaft des `image`-Movieclips animieren.

Sie benötigen dazu eine boolesche Variable, die es Ihnen erlaubt, den momentanen Zustand (skaliert oder nicht skaliert) zu speichern und die Eigenschaften der Animation entsprechend anzupassen. Fügen Sie noch vor der `onPress`-Funktion in Zeile 3 folgende Codezeile ein:

```
image.isScaled = false;
```

Direkt danach legen Sie zwei Variablen für den maximalen und den minimalen Skalierungswert fest:

```
image.scaleMax = 100;
image.scaleMin = image._xscale = image._yscale = 50;
```

In der zweiten Zeile weisen Sie den Eigenschaften `_xscale` und `_yscale` von `image` bereits den `scaleMin`-Wert zu. Das Bild ist also zu Beginn um die Hälfte verkleinert.

Erweitern Sie jetzt die `onPress`-Funktion folgendermaßen:

```
1: image.onPress = function() {
2:     var isDoubleClick:Boolean = DoubleClick.
      isDoubleClick(this);
3:     if(isDoubleClick){
4:         this.xscaleTween.stop();
5:         this.yscaleTween.stop();
6:         var scale:Number = this.isScaled ? this.scaleMin :
      this.scaleMax;
7:         this.xscaleTween = new Tween(this,'_xscale',Back.
      easeInOut,this._xscale,scale,.7,true);
8:         this.yscaleTween = new Tween(this,'_yscale',Back.
      easeInOut,this._xscale,scale,.7,true);
9:         this.isScaled = !this.isScaled;
10:    }
11: };
```

In Zeile 3 wird abgefragt, ob es sich um einen Doppelklick handelt. Ist dies der Fall, wird die laufende Animation gestoppt, die ab Zeile 7 gestartet wird. Dies geschieht nur für den Fall, dass »wie wild« auf das Bild geklickt wird und so mehrere Doppelklicks schnell hintereinander erzeugt würden. Jeder dieser Doppelklicks würde dann eine Animation auslösen. In Zeile 6 wird eine Variable `scale` je nach Zustand der Variable `isScaled` mit dem Wert von `scaleMax` oder `scaleMin` initialisiert. Dieser Wert wird nun bei der Initialisierung von `Tween` in den Zeilen 7 und 8 als Zielwert verwendet. Zum Schluss müssen Sie noch `isScaled` in dessen gegenteiligen Wert setzen.

Führen Sie nun den Flash-Film mit `[Strg]/[⌘]+[↩]` aus, und schauen Sie sich das Ergebnis an. Das Bild ist anfänglich kleiner skaliert. Doppelklicken Sie auf das Bild, wird es größer skaliert. Wenn Sie erneut auf das Bild dop-

Video-Training

Was sind Funktionen eigentlich genau? Schauen Sie doch mal in die Video-Lektion 2.1 auf der Buch-DVD rein.

Schritt 12

Film testen

pelklicken, wird es wieder kleiner skaliert. Voilà, Sie sind stolzer Programmierer einer Doppelklick-Anwendung!

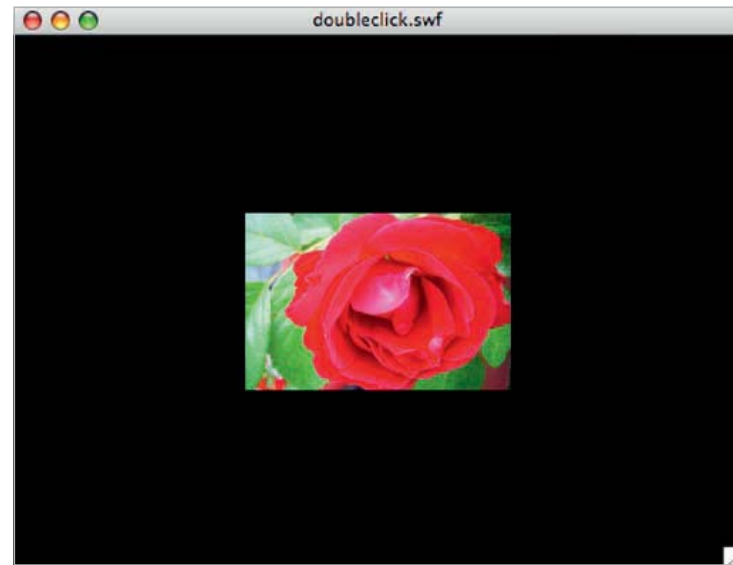


Abbildung 5 ►
Beim Aufrufen des Flash-Films ist das Bild klein.

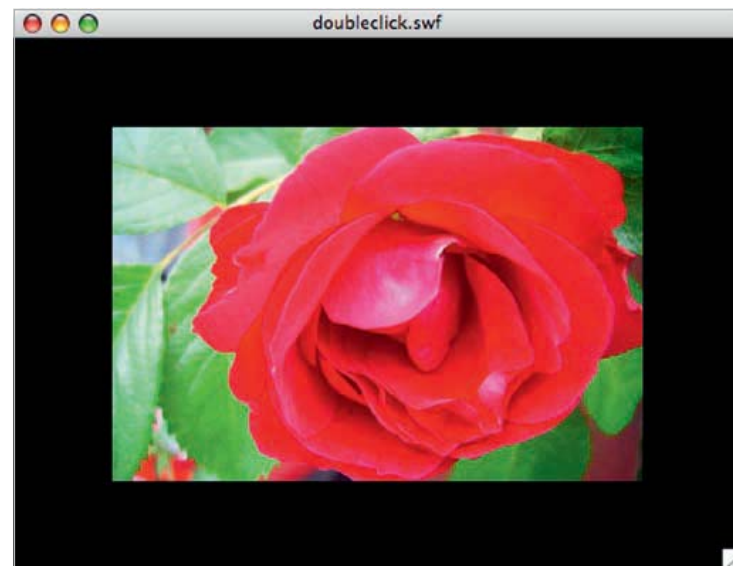


Abbildung 6 ►
Per Doppelklick können Sie es vergrößern.

Tooltipps

Programmieren einer eigenen Tooltipp-Klasse

Vermutlich kennen Sie Tooltipps aus dem Internet oder von Desktopanwendungen als nützliches Hilfsmittel. Auch bei Flash-Anwendungen können diese hilfreich sein. In diesem Workshop erstellen Sie eine eigene, wieder verwertbare Tooltipp-Klasse, die Sie künftig in beliebigen Flash-Projekten verwenden können.

Zielsetzungen:

Eine Tooltipp-Sprechblase mit ActionScript zeichnen

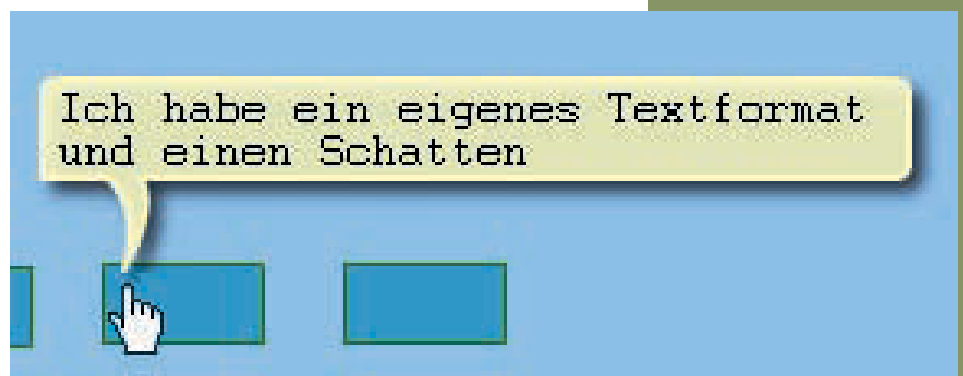
Benötigte Parameter übergeben und verarbeiten

DropShadowFilter anwenden

[Ergebnis: 23_TOOLTIPPS •
TOOLTIP.SWF]

```
import mx.transitions.Tween;
import mx.transitions.easing.*;
import flash.filters.DropShadowFilter;

class Tooltip{
    var ttWidth:Number;
    var ttHeight:Number;
    var ttColor:Number;
    var ttAlpha:Number;
    var ttText:String;
    var ttTextFormat:TextFormat;
    var linkOnRollOver:Function;
    var linkOnRollOut:Function;
    var shadowEnabled:Boolean;
    public function Tooltip(link:MovieClip, ttText:String,
        ttColor:Number, ttAlpha:Number){
        this.ttText = ttText||"Kein Text vorhanden";
        this.ttColor = ttColor||0xeeeeee;
        this.ttAlpha = ttAlpha||90;
        initToolTip(link);
    }
    private function initToolTip(link:MovieClip):Void{
        var scope:Object = this;
        link.trackAsMenu = true;
        link.onRollOver = link.onDragOver = function(){
```



Schritt 1 Vorbereitungen

Schritt 2 Rechteck als Platzhalter- Movieclip

Legen Sie auf Ihrer Festplatte einen Ordner mit dem Namen TOOLTIPP an. In diesem Ordner werden alle benötigten Dateien gespeichert.

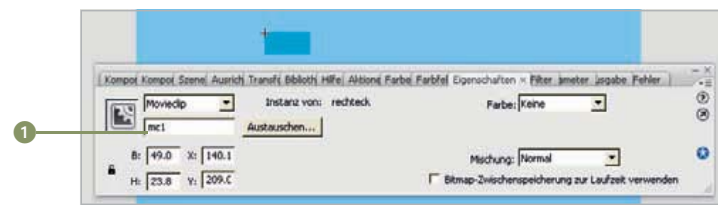
In diesem Workshop beinhaltet die ActionScript-Datei sämtliche Funktionalität. Die Flash-Datei dient nur Testzwecken. Im praktischen Einsatz verwenden Sie die Flash-Datei Ihres jeweiligen Projektes.

Legen Sie ein neues Flash-Dokument (ActionScript 2) an, und speichern Sie es unter dem Namen TOOLTIPP.FLA. Wählen Sie das Rechteckwerkzeug , und zeichnen Sie auf der Bühne ein beliebiges Rechteck.

Markieren Sie das eben erstellte Rechteck, und wählen Sie nach Rechtsklick bzw. **Ctrl**-Klick im erscheinenden Kontextmenü den Eintrag **IN SYMBOL KONVERTIEREN**. Wählen Sie als Typ die Option **MOVIECLIP**, und geben Sie den Namen »Rechteck« ein.

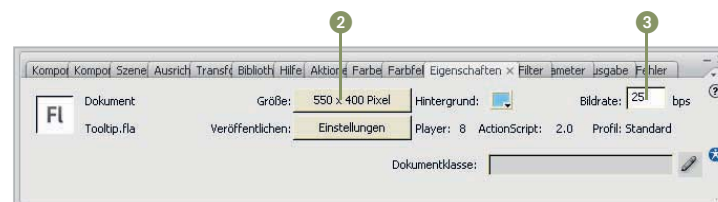
Geben Sie dem neu erstellten Symbol im Eigenschaftenfenster den Instanznamen »mc1« **1**. Dieser Movieclip dient als Platzhalter für einen beliebigen Movieclip Ihrer Benutzeroberfläche, dem ein Tooltipp zugeordnet werden soll.

Abbildung 1 ►
Das Rechteck dient nur als Platzhalter. In Ihren Projekten können Sie einem beliebigen Movieclip den Tooltipp hinzufügen.



Wählen Sie eine Bühnengröße von 550 x 400 Pixel **2**. Als BILDRATE **3** sollten Sie mindestens 25 bps wählen, da der Tooltipp der Mausbewegung folgen wird. Eine niedrige Bildrate würde zu einer ruckeligen Verfolgungsbewegung führen.

Abbildung 2 ►
Im Eigenschaftenfenster legen Sie Größe und Bildrate der Flash-Datei fest.



Schritt 3 ActionScript-Datei anlegen

Wählen Sie **DATEI • NEU • ACTIONSCRIPT-DATEI**. Speichern Sie die Datei unter dem Namen TOOLTIP.AS in demselben Ordner wie die Flash-Datei. Geben Sie folgende Codezeilen ein:


```

1: import mx.transitions.Tween;
2: import mx.transitions.easing.*;
3: import flash.filters.DropShadowFilter;
4: class Tooltip{
5: }

```

In Zeile 1 wird die Tween-Klasse importiert, die Tweenings per ActionScript ermöglicht. Sie werden ein Tweening verwenden, um den Tooltip der Maus folgen zu lassen. Da eine weiche Bewegung (engl.: Easing) schöner ist als eine abrupte, importieren Sie in der zweiten Zeile alle im Ordner MX/TRANSITIONS/EASING befindlichen Animationstypen. Zu guter Letzt importieren Sie in Zeile 3 noch den DropShadowFilter.

Als Nächstes deklarieren Sie die Klassenvariablen. Ergänzen Sie unterhalb der öffnenden geschweiften Klammer folgenden Code:

```

var ttWidth:Number;
var ttHeight:Number;
var ttColor:Number;
var ttAlpha:Number;
var ttText:String;
var ttTextFormat:TextFormat;
var linkOnRollOver:Function;
var linkOnRollOut:Function;
var shadowEnabled:Boolean;

```

Fügen Sie unterhalb der Variablendeklarationen folgende Codezeilen ein:

```

1: public function Tooltip(link:MovieClip, ttText:String,
   ttColor:Number, ttAlpha:Number){
2:     this.ttText = ttText||"Kein Text vorhanden";
3:     this.ttColor = ttColor||0xeeeeee;
4:     this.ttAlpha = ttAlpha||90;
5:     initToolTip(link);
6: }

```

Als Parameter erwartet der Konstruktor den Movieclip, dem der Tooltip zugeordnet werden soll, sowie den anzuzeigenden Text, die Hintergrundfarbe des Tooltips und den Alphawert des Tooltips.

Schritt 4

Klassenvariablen deklarieren

Schritt 5

Der Konstruktor

In den Zeilen 2 bis 4 werden den Variablen Standardwerte zugeordnet, sofern diese nicht beim Konstruktoraufbau angegeben wurden. So ist es im Prinzip möglich, nur einen Movieclip zu übergeben. Dieser erhält dann einen Tooltip mit Standardwerten und dem Hinweis »Kein Text vorhanden«, womit der Zweck der Klassenvariable `ttText` auch bereits erklärt wäre.

Zum Schluss rufen Sie die Funktion `initToolTip()` mit dem Parameter `link` auf. Diese Funktion müssen Sie natürlich noch definieren, bevor etwas geschieht.

Schritt 6

Grundfunktionalität des Tooltips implementieren

Die Grundfunktionalität des Tooltips wird in der Funktion `initToolTip()` festgelegt.

In der Funktion `initToolTip()` wird im Wesentlichen festgelegt, dass beim Bewegen der Maus über den Movieclip eine Funktion zum Erstellen eines Tooltips ausgeführt werden soll. Außerdem wird festgelegt, dass beim Bewegen der Maus aus dem Movieclip heraus der Tooltip wieder verschwinden soll.

Ergänzen Sie folgende Funktion unterhalb des Konstruktors:

```
1: private function initToolTip(link:MovieClip):Void{
2:     var scope:Object = this;
3:     link.trackAsMenu = true;
4:     link.onRollOver = link.onDragOver = function(){
5:         scope.drawToolTip(this);
6:         scope.linkOnRollOver();
7:     }
8:     link.onRollOut = link.onDragOut = function(){
9:         removeMovieClip(this._parent.tooltip);
10:        scope.linkOnRollOut();
11:    }
12: }
```

In dieser Funktion gibt es nur einen Parameter vom Typ Movieclip – der Movieclip, der den Tooltip bekommt.

In Zeile 3 wird über die `trackAsMenu`-Eigenschaft prinzipiell festgelegt, dass der Movieclip das Loslassen der Maustaste als Ereignis empfangen kann, unabhängig davon, ob die Maustaste über dem Movieclip oder außerhalb des Movieclips gedrückt wurde.

In Zeile 4 kommt diese Eigenschaft dann auch zum Einsatz. Es wird eine Funktion definiert, die sowohl beim Ziehen der Maus über den Movieclip mit gedrückter linker Maustaste (`onDragOver`) wie auch mit nicht gedrück-

ter linker Maustaste (`onRollOver`) ausgeführt wird. In der Funktion selbst wird zunächst eine weitere Funktion aufgerufen, die den Tooltip zeichnet. Mehr dazu im nächsten Schritt.

Schließlich wird noch die Variable `linkOnRollOver` als Funktion aufgerufen. Dies schafft Abhilfe für das Problem, dass die `onRollOver`- bzw. `onDragOver`-Prozeduren bereits »verbraucht« sind und so in Verbindung mit unserer Tooltip-Klasse kein weiteres Verhalten hierfür festgelegt werden könnte. Mehr dazu erfahren Sie in Schritt 10, »Setter-Methoden«.

Anschließend wird nach demselben Schema das Verhalten für das Ziehen der Maus aus dem Movieclip heraus festgelegt (Zeile 8 bis 11).

Nun ist es an der Zeit, den eigentlichen Tooltip zu erstellen. Hierzu verwenden wir die Funktion `drawToolTip()`. Ergänzen Sie folgende Codezeilen unter der Funktion `initToolTip()`:

```
1: private function drawToolTip(link:MovieClip):Void{
2:     var ttHandler:MovieClip = link._parent.
        createEmptyMovieClip("toolTip", link._parent.
        getNextHighestDepth());
3:     var textHandler:TextField = ttHandler.createTextField
        ("toolTipText", ttHandler.getNextHighestDepth(), 0, 0 , 1,
        1);
4:     textHandler.text = ttText;
5:     textHandler.autoSize = true;
6:     textHandler.selectable = false;
7:     if(ttTextFormat==undefined){
8:         ttTextFormat = new TextFormat();
9:         ttTextFormat.font = "Verdana";
10:        ttTextFormat.size = 12;
11:        ttTextFormat.color = 0x000000;
12:    }
13:    textHandler.setTextFormat(ttTextFormat);
14:    ttWidth=ttHandler._width;
15:    ttHeight=ttHandler._height;
16:    textHandler._y=-ttHeight-30;
17:    textHandler._x=-23;
18:    if(ttWidth<30){
19:        ttWidth=30;
20:        textHandler._x=-8-ttHandler._width/2;
21:    }
22:    drawBackGround(ttHandler, link);
23: }
```

Funktionen als Variablen?

In ActionScript ist auch eine Funktion eine Variable vom Typ `Function`. So können Sie mit variablen Funktionen arbeiten. Die hier verwendeten Callback-Funktionen sind ein typisches Anwendungsbeispiel.

Schritt 7 Tooltip-Movieclip zeichnen

Zunächst erstellen Sie mit `createEmptyMovieClip()` einen leeren Movieclip. Dieser Movieclip ist der spätere Tooltip, der hier natürlich noch über keinerlei Grafik verfügt und somit auch nicht sichtbar ist.

Der Movieclip wird innerhalb des dem übergebenen Movieclips `link` übergeordneten Elements (`_parent`) erstellt. Im Falle unserer simplen Test-Flash-Datei ist das die Bühne selbst – je nach Projektstruktur kann das jedoch auch anders sein. Durch die Zuweisung zum übergeordneten Element wird sichergestellt, dass die Tooltips nicht von anderen Movieclips auf der gleichen Ebene überdeckt werden. Während es sich bei `ttHandler` um eine lokale Variable handelt, die nur innerhalb der Funktion verwendet werden kann, steht der Parameter `toolTip` für den globalen Instanznamen des neuen Movieclips, der innerhalb der gesamten Klasse und sogar außerhalb der Klasse verwendet werden kann. Sie haben diesen Namen bereits in der Funktion `initToolTip()` verwendet:

```
removeMovieClip(this.toolTip);
```

In den Zeilen 4 bis 13 werden dem Textfeld der Tooltip-Text und eine Formatierung zugewiesen.

Für die Breite und Höhe stehen die Klassenvariablen `ttWidth` (Breite) und `ttHeight` (Höhe) bereit (vergleiche die Schritte 4 und 5). Da Sie per `autoSize` die Breite und Höhe des Textfeldes ermittelt haben (Zeile 5), können die so ermittelten Werte jetzt den Klassenvariablen zugewiesen werden (Zeilen 14 und 15). Das Zeichnen der Sprechblase ist in eine gesonderte Funktion ausgelagert, die die Variablen `ttWidth` und `ttHeight` verarbeitet.

Außerdem kann das Textfeld nun, da seine Größe bekannt ist, richtig positioniert werden (Zeilen 16 bis 23), wobei eine Breite von weniger als 30 Pixel als Sonderfall abgefangen wird.

Das Zeichnen des Hintergrundes geschieht in einer eigenen Funktion, die in Zeile 22 aufgerufen wird.

Andere Positionierungen

Die pixelgenauen Angaben sowie das Abfangen einer sehr kleinen Breite haben einen rein gestalterischen Hintergrund. Sie können mit diesen Werten ruhig ein wenig experimentieren, allerdings sollten Sie sich damit noch bis zum Ende von Schritt 11 gedulden.

Schritt 8 Sprechblase zeichnen

Nun wollen wir die Funktion für den Hintergrund implementieren. Ergänzen Sie folgenden Code unter der in Schritt 7 hinzugefügten Funktion:

```
1: private function drawBackGround(ttHandler:MovieClip,  
   link:MovieClip):Void{  
2:     ttHandler._x=ttHandler._parent._xmouse;  
3:     ttHandler._y=ttHandler._parent._ymouse;  
4:     ttHandler.lineStyle(2, ttColor, 100);
```

```

5:         ttHandler.moveTo(0, -2);
6:         ttHandler.beginFill(this.ttColor, ttAlpha);
7:         ttHandler.curveTo(5,-20,-5,-30);
8:         ttHandler.lineTo(-25, -30);
9:         ttHandler.curveTo(-25, -30, -25, -35);
10:        ttHandler.lineTo(-25, -25-ttHeight);
11:        ttHandler.curveTo(-25, -30-ttHeight,-20,
        -30-ttHeight);
12:        ttHandler.lineTo(ttWidth-25, -30-ttHeight);
13:        ttHandler.curveTo(ttWidth-20, -30-ttHeight,ttWidth-20,
        -25-ttHeight);
14:        ttHandler.lineTo(ttWidth-20, -35);
15:        ttHandler.curveTo(ttWidth-20, -30,ttWidth-25, -30);
16:        ttHandler.lineTo( 5, -30);
17:        ttHandler.curveTo( 10, -20, 0, -2);
18:        ttHandler.endFill();
19:        if(shadowEnabled) drawShadow(ttHandler);
20:        ttHandler.cacheAsBitmap=true;
21:        ttHandler.onEnterFrame = function(){
22:            var xPosT:Tween = new Tween(this, "_x", Elastic.
                easeOut, this._x, this._parent._xmouse, 25, false);
23:            var yPosT:Tween = new Tween(this, "_y", Elastic.
                easeOut, this._y, this._parent._ymouse, 25, false);
24:        }
25:    }

```

Diese Funktion sieht komplizierter aus als sie ist. In den Zeilen 2 und 3 wird die Startposition des Tooltips festgelegt. Das Positionieren in Echtzeit, also das Verfolgen der Mausbewegung, geschieht in der `onEnterFrame`-Methode in den Zeilen 21 bis 24 mit Hilfe der Tween-Klasse. Das Zuweisen der Startposition verhindert, dass der Tooltip aus der linken oberen Ecke der Bühne »angeflogen« kommt.

In den Zeilen 6 bis 18 geschieht das eigentliche Zeichnen. Näheres zu den hier verwendeten Funktionen `moveTo()` und `lineTo()` können Sie der Flash-Hilfe entnehmen. Hier werden die Höhe (`ttWidth`) und Breite (`ttHeight`) verarbeitet. Außerdem werden einige fixe Zahlen verwendet, die allesamt die Darstellung der Sprechblase verfeinern. Wenn Sie wollen, experimentieren Sie ruhig ein wenig mit diesen Werten und sehen Sie sich die verschiedenen Ergebnisse an.

In Zeile 19 wird gegebenenfalls die Funktion zum Hinzufügen eines Schlagschattens ausgeführt (siehe Schritt 9). Die `cacheAsBitmap`-Anweisung in Zeile 20 sorgt für eine bessere Performance.

this._parent._xmouse

Mit `_xmouse` können die Koordinaten der Maus abgefragt werden. Hier muss jedoch zwischen Bühnenkoordinaten und den Koordinaten eines Movieclips unterschieden werden – Movieclips haben ein eigenes Koordinatensystem. Da hier die Mausposition in Bezug auf die Bühne abgefragt werden soll, muss `_parent` verwendet werden – das Elternelement des Tooltips ist die Bühne.

Schritt 9 Schlagschatten hinzufügen

Um das Ergebnis testen zu können, muss noch die Funktion `drawShadow()` definiert werden. Fügen Sie unter die eben definierte Funktion `drawBackground()` folgenden Code ein:

```
1: private function drawShadow(ttHandler:MovieClip):Void{
2:     var filter:DropShadowFilter = new DropShadowFilter(5, 45,
3:         0x000000, .8, 5, 5, 1, 3, false, false, false);
4:     var filterArray:Array = new Array();
5:     filterArray.push(filter);
6:     ttHandler.filters = filterArray;
7: }
```

Die Funktion fügt einem beliebigen Movieclip einen Schlagschatten mit fixen Attributen hinzu. Dieser wird in Zeile 2 erstellt und in den Zeilen 3 bis 5 dem Tooltipp-Movieclip zugewiesen.

Schritt 10 Setter-Methoden

Zu guter Letzt muss noch der Zugriff auf diejenigen Variablen bereitgestellt werden, die weder im Konstruktor übergeben noch dynamisch ermittelt werden: `ttTextFormat`, `linkOnRollOver`, `linkOnRollOut` und `shadowEnabled`.

Dies geschieht über Setter-Methoden. Ergänzen Sie folgende Funktionen unter der zuletzt eingefügten:

```
1: public function set _textformat(ttTextFormat:TextFormat):
2:     Void{
3:         this.ttTextFormat = ttTextFormat;
4:     }
5: public function set _onrollover(linkOnRollOver:Function):
6:     Void{
7:         this.linkOnRollOver = linkOnRollOver;
8:     }
9: public function set _onrollout(linkOnRollOut:Function):
10:    Void{
11:        this.linkOnRollOut = linkOnRollOut;
12:    }
13: public function set _shadow(shadowEnabled:Boolean):Void{
14:     this.shadowEnabled = shadowEnabled;
15: }
```

Die Eigenschaften `_onmouseover` und `_onmouseout` stehen für die Funktionen, die beim Überfahren bzw. Herausfahren mit der Maus ausgeführt werden sollen (vergleiche Schritt 6). Schließlich wird in Zeile 10 noch die Setter-Methode für den Schlagschatten implementiert. Die Klasse ist fertig – Zeit für einen gründlichen Test.

Eine Setter-Funktion wird durch das Schlüsselwort `set` gekennzeichnet und ermöglicht den Zugriff auf die Eigenschaften eines Objekts zur Laufzeit. So wird beispielsweise in Schritt 10, Zeile 1 die Übergabe eines eigenen Textformats ermöglicht. Dieses kann dann einfach so gesetzt werden:

```
Tooltip_instanz._textformat = textformat_instanz;
```

Kopieren Sie den in Schritt 2 erstellten Rechteck-Movieclip nun dreimal, und vergeben Sie als Instanznamen »mc1« (bereits erfolgt), »mc2«, »mc3« und »mc4«. Legen Sie dann folgendes Skript auf Bild 1 der Hauptzeitleiste:

```
1: var tt1:Tooltip = new Tooltip(mc1, "Ich habe nur den Text als
   Parameter");
2: var tt2:Tooltip = new Tooltip(mc2, "Ich habe einen Text und
   \n eine Farbe und bin transparent", 0xffffcc, 60);
3: var tt3:Tooltip = new Tooltip(mc3, "Ich habe ein eigenes
   Textformat \n und einen Schatten", 0xffffcc);
4: var tf:TextFormat = new TextFormat();
5: tf.font = "Courier";
6: tf.size = 12;
7: tf.color = 0x000000;
8: tt3._shadow = true;
9: tt3._textformat = tf;
10: var tt4:Tooltip = new Tooltip(mc4, "Ich nutze die
    \nRollOver-/RollOut Callbacks", 0xffffcc);
11: tt4._onmouseover=function(){
12:     mc4._alpha=30;
13: }
14: tt4._onmouseout=function(){
15:     mc4._alpha=100;
16: }
```

Schritt 11

Anwenden der Tooltipp-Klasse

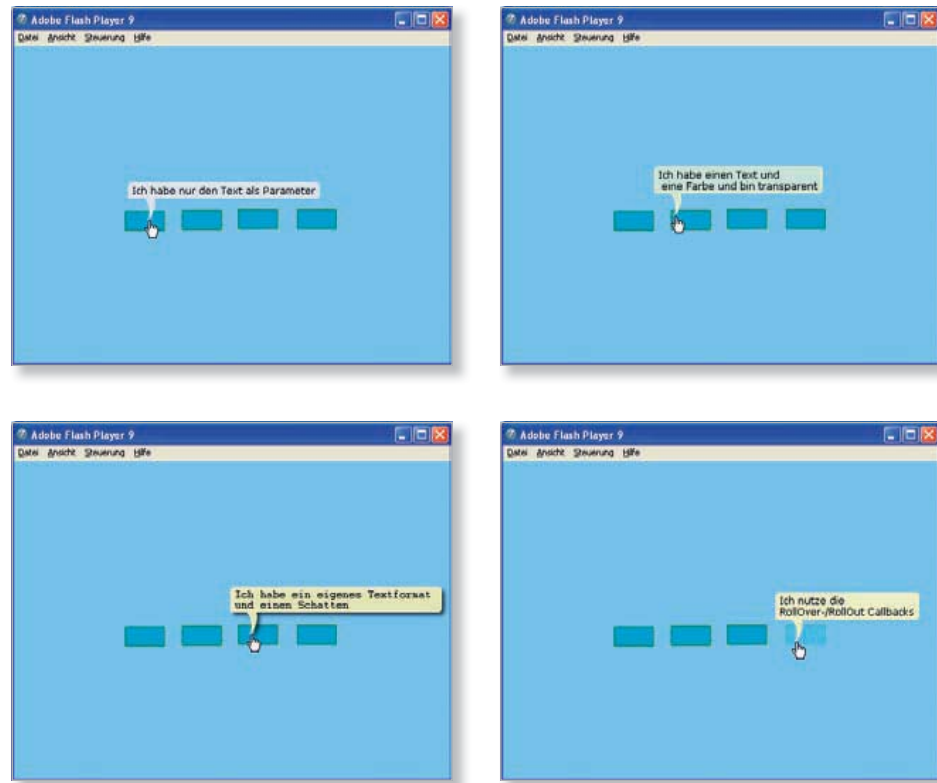


Abbildung 4 ►
Vier individuell eingestellte
Tooltips

Dateiupload

Dateien auf einen Server übertragen

In diesem Workshop werden Sie einen Dateiupload in ActionScript 2 unter Zuhilfenahme der FileReference-Klasse erstellen. Hierbei handelt es sich um eine eigene swf-Datei, die als Dialogfenster in ein beliebiges Flash-Projekt geladen werden kann.

Zielsetzungen:

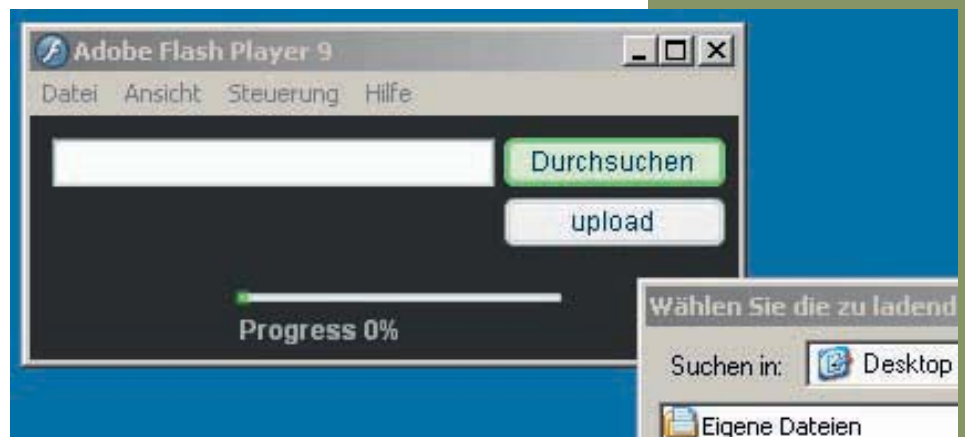
Mit Komponenten arbeiten

FileReference-Klasse einsetzen

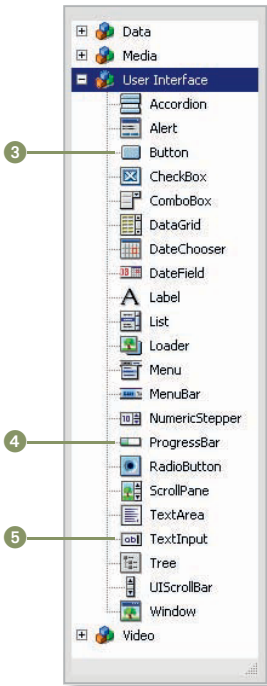
[Ergebnis: 24_DATEIUPLOAD

• FILEUPLOAD.SWF]

```
import flash.net.FileReference;
var url:String = "upload.php";
var fileReference:FileReference = new FileReference();
var listener:Object = new Object();
listener.onSelect = function(file:FileReference):Void{
    tf_filename.text = file.name;
}
listener.onProgress = function(file:FileReference,
bytesLoaded:Number, bytesTotal:Number):Void {
    progressBar.setProgress(bytesLoaded, bytesTotal);
}
fileReference.addListener(listener);
btn_search.onRelease = function():Void{
    fileReference.browse();
}
btn_upload.onRelease = function():Void{
    fileReference.upload(url);
}
```



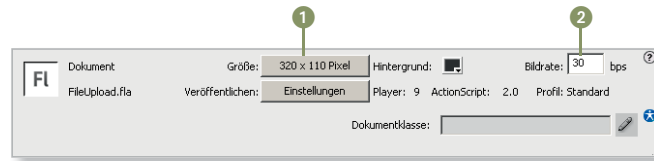
Schritt 1 Anlegen der Flash-Datei



▲ **Abbildung 2**
Komponenten in Flash CS3

Abbildung 3 ►
Mit den Komponenten können Sie schnell und einfach User Interfaces erstellen.

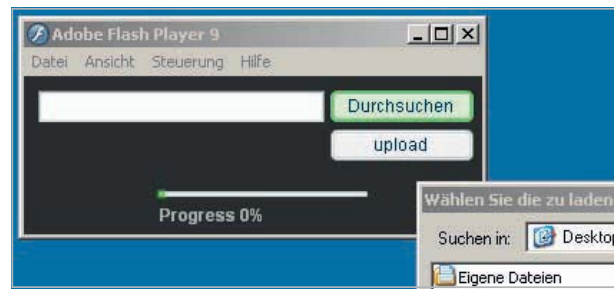
Legen Sie ein neues Flash-Dokument an (ActionScript 2). Wählen Sie eine Bühnengröße von 320x110 Pixel **1** und eine BILDRATE **2** von 30 bps – das entspricht der Größe des Dialogfensters.



▲ **Abbildung 1**
Legen Sie die Dokumenteigenschaften fest.

Öffnen Sie das Komponenten-Fenster über **Strg**/**⌘** + **F7**, und ziehen Sie die Komponenten TEXTINPUT **5**, PROGRESSBAR **4** und BUTTON **3** aus dem Bereich USER INTERFACE auf die Bühne. Erstellen Sie eine zweite Instanz der Button-Komponente, und beschriften Sie die Buttons mit »Durchsuchen« und »upload«.

Ordnen Sie die Komponenten so an, dass sie ein ansehnliches Dialogfenster ergeben.



Vergeben Sie nun folgende Instanznamen: »tf_filename« für das Textfeld, »btn_search« und »btn_upload« für die Buttons und »progressBar« für die Statusbalken-Instanz.

Achten Sie außerdem darauf, dass die Parameter (FENSTER • EIGENSCHAFTEN • PARAMETER) EDITABLE **6** und PASSWORD **7** der TextInput-Instanz auf »false« gestellt sind – das Textfeld soll weder ein Passwort enthalten, noch direkt editierbar sein.

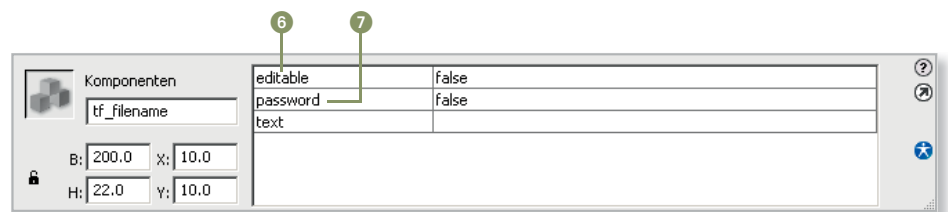


Abbildung 4 ►
Parameter einstellen

Mit der `FileReference`-Klasse können Sie auf recht einfache Art Daten von einem Client auf einen Server übertragen. Ein `FileReference`-Objekt stellt einen Dialog zur Verfügung, mit dem eine Datei auf der Festplatte ausgewählt werden kann. Über diese Datei liegen dann Informationen in Form von Eigenschaften der `FileReference`-Instanz vor. Außerdem stellt die Klasse nützliche Funktionen bereit, etwa zum Hochladen einer Datei.

Um mit der `FileReference`-Klasse arbeiten zu können, muss diese importiert und dann ein Objekt erstellt werden. Außerdem wird für das Hochladen eine Pfadangabe benötigt. Diese muss auf ein Skript verweisen, das den Upload serverseitig übernimmt. Fügen Sie eine Ebene »script« in Ihre Flash-Datei ein, und legen Sie folgenden Code auf Bild 1 der Hauptzeileiste:

```
import flash.net.FileReference;
var url:String = "upload.php";
var fileReference:FileReference = new FileReference();
```

Beachten Sie, dass sich bei dieser relativen Pfadangabe das Skript `UPLOAD.PHP` in demselben Verzeichnis wie die `swf`-Datei befinden muss. Außerdem muss sich dieses Verzeichnis im `htdocs`-Verzeichnis eines lokalen Servers befinden.

Falls Sie keine Lust haben, einen lokalen Server einzurichten, können Sie auch folgende URL verwenden: <http://www.flashfastforward.de/upload.php>. Auch hier befindet sich das unten abgedruckte Skript.

Das `FileReference`-Objekt braucht einen Listener, um auftretende Ereignisse auch zu bemerken. Ergänzen Sie folgenden Code ganz unten in Ihrem Skript:

```
1: var listener:Object = new Object();
2: listener.onSelect = function(file:FileReference):Void{
3:     tf_filename.text = file.name;
4: }
5: listener.onProgress = function(file:FileReference,
6:     bytesLoaded:Number, bytesTotal:Number):Void {
7:     progressBar.setProgress(bytesLoaded, bytesTotal);
8: }
8: fileReference.addListener(listener);
```

Schritt 2

Die `FileReference`-Klasse einsetzen

Relative und absolute Pfadangaben

Eine absolute Pfadangabe gibt eindeutig den Speicherort einer Datei an. Mit ihr kann auf entfernte Rechner zugegriffen werden. Sie beinhaltet daher den kompletten Pfad der Datei. Eine relative Pfadangabe hingegen beinhaltet den Pfad ausgehend von einem bestimmten Verzeichnis. Sie kann nur verwendet werden, um auf Dateien auf dem lokalen Rechner (der auch ein Server sein kann) zuzugreifen. Die Angabe eines Dateinamens ohne Pfad führt dazu, dass die jeweilige Datei ausschließlich in demselben Verzeichnis gesucht wird – und dort natürlich auch vorhanden sein sollte.

Schritt 3

Listener hinzufügen

In Zeile 1 wird ein Listener-Objekt erstellt. In Zeile 2 wird festgelegt, dass das Textfeld den Namen der ausgewählten Datei anzeigen soll. In den Zeilen 5 bis 6 wird festgelegt, dass der Statusbalken immer den aktuellen Ladestatus anzeigen soll. In Zeile 8 schließlich wird der Listener dem File-Reference-Objekt zugeordnet.

Schritt 4 Die Funktionalität der Buttons

Nun brauchen nur noch die Buttons ihre Funktionalität. Ergänzen Sie folgenden Code, wieder ganz am Ende des Skripts:

```
1: btn_search.onRelease = function():Void{
2:     fileReference.browse();
3: }
4: btn_upload.onRelease = function():Void{
5:     fileReference.upload(url);
6: }
```

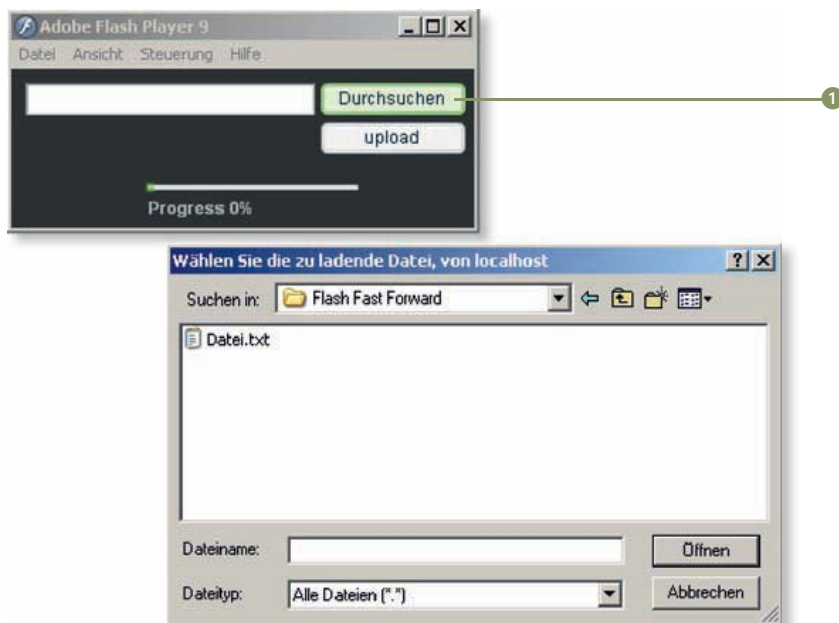
Dem DURCHSUCHEN-Button wird die FileReference-Methode `browse()` zugeordnet (Zeile 2). Diese startet einen Dateibrowser, der die Auswahl einer Datei ermöglicht. Der UPLOAD-Button erhält als Funktionalität die FileReference-Methode `upload()`, die den Upload zur angegebenen URL startet.

Schritt 5 PHP-Skript anlegen

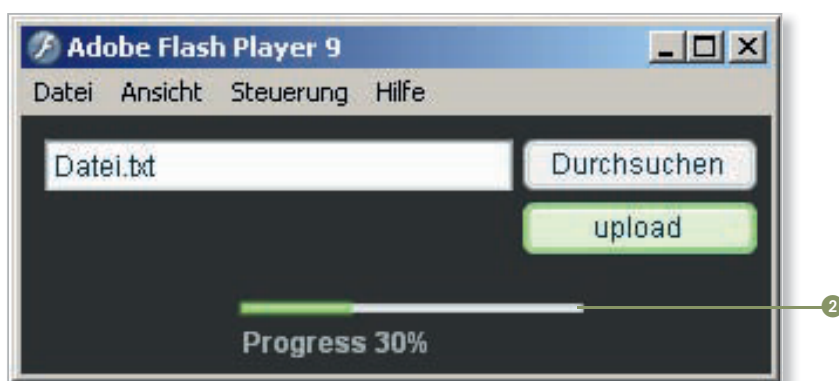
Zu guter Letzt benötigen Sie noch ein entsprechendes PHP-Skript:

```
<?php
$folder = <uploads>;
$uploadfile = "$folder/" . basename( $_FILES['Filedata']['name']
);
if ( move_uploaded_file( $_FILES['Filedata']['tmp_name'] ,
$uploadfile ) ) {
    echo( '1 ' . $_FILES['Filedata']['name']);
}else{
    echo( '0');
}
?>
```

Dieses Skript speichert die hochgeladenen Dateien in einem Ordner `UP-LOAD`. Dieser muss natürlich zuvor auf dem Server angelegt werden.



◀ **Abbildung 5**
Die Datei wird über einen Klick auf DURCHSUCHEN ① ausgewählt.



◀ **Abbildung 6**
Anschließend zeigt der Progress-Balken ② den Lade-status an.

Individuelle Kontextmenüs

Programmieren einer eigenen Kontextmenü-Klasse

```
class CMenu{
    private var cm:ContextMenu;
    public function CMenu(showDefault:Boolean){
        cm = new ContextMenu();
        if(!showDefault) cm.hideBuiltInItems();
    }
    public function addBaseObject(bo: MovieClip){
        bo.menu = cm;
    }
    public function addCMenuItem(itemName:String,
        menuText:String, callBack:Function,
        seperator:Boolean):Boolean{
        for(var i:Number=0; i<cm.customItems.length; i++){
            if((cm.customItems[i] ==
```

Sicherlich ist Ihnen schon aufgefallen, dass ein Rechtsklick auf einen beliebigen Flash-Film meist ein Standard-Kontextmenü erscheinen lässt. Da auch der Flash Player eine Anwendung ist, bezieht sich dieses Kontextmenü auf den Flash Player selbst, und nicht auf den Inhalt des dargestellten Flash-Films. Das muss aber nicht so sein. So wie Sie in Flash fast alles manipulieren können, können Sie auch ein eigenes, auf Ihre Flash-Applikation zugeschnittenes Kontextmenü erstellen. Wie das geht, erfahren Sie in diesem Workshop.

Zielsetzungen:

Das Standard-Kontextmenü anpassen

ActionScript 2-Klasse zur Verwaltung eigener Kontextmenüs erstellen

[Ergebnis: 25_ KONTEXTMENUE • KONTEXTMENUE.SWF]

Adobe Flash Player 9

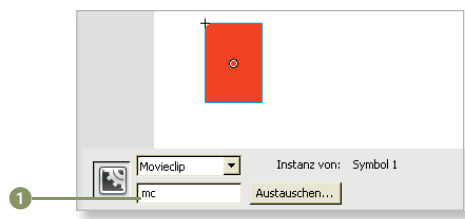
Datei Ansicht Steuerung Hilfe



Im ersten Schritt legen Sie einen neuen Ordner auf Ihrer Festplatte an und nennen ihn KONTEXTMENUE. Hier werden Sie die von Ihnen erstellten Beispiele zu diesem Workshop abspeichern.

Die Flash-Datei dient hier nur zu Testzwecken – schließlich ist das Kontextmenü, wie der Name schon sagt, abhängig vom Kontext. In diesem Workshop werden Sie es lediglich mit beispielhafter Funktionalität ausstatten. Bühnengröße und Bildrate sind deshalb hier unerheblich, weshalb Sie einfach die Standardeinstellungen übernehmen können.

Erstellen Sie ein neues Flash-Dokument (AS2). Markieren Sie Bild 1 der Hauptzeitleiste, und zeichnen Sie ein beliebiges Rechteck auf der Bühne. Markieren Sie es, und öffnen Sie mit einem Rechtsklick bzw. **Ctrl**-Klick das Kontextmenü. Wählen Sie den Eintrag **IN SYMBOL KONVERTIEREN**. Als **TYP** wählen Sie **MOVIECLIP**, als Name können Sie den Standardnamen »Symbol 1« verwenden. Vergeben Sie anschließend im Eigenschaftenfenster den Instanznamen »mc« 1.



Mit Hilfe dieses Movieclips werden wir später veranschaulichen, wie Sie verschiedenen Elementen Ihrer Flash-Anwendung eigene Kontextmenüs zuweisen können – hier dem Movieclip und der Bühne.

Speichern Sie die Datei unter dem Namen KONTEXTMENUE.FLA ab.

Erstellen Sie eine ActionScript 2-Datei (DATEI • NEU • ACTIONSCRIPT-DATEI), und speichern Sie diese in demselben Verzeichnis unter dem Namen CMENU.AS. Geben Sie folgenden Code ein:

```
1: class CMenu{
2:     var cm:ContextMenu;
3:     public function CMenu(showDefault:Boolean){
4:         cm = new ContextMenu();
5:         if(!showDefault) cm.hideBuiltInItems();
6:     }
7: }
```

Schritt 1

Vorbereitungen

Schritt 2

Anlegen der Flash-Datei

◀ **Abbildung 1**
Der Movieclip erhält im Eigenschaftenfenster einen Instanznamen.

Schritt 3

ActionScript-Datei anlegen

Als einzige Klassenvariable wird eine Variable `cm` vom Typ `ContextMenu` deklariert. `ContextMenu` ist eine Standard-Flash-Klasse, auf die Sie hier zurückgreifen. Mehr darüber lesen Sie in Schritt 5. Als Konstruktorparameter wird eine boolesche Variable übergeben, die angibt, ob die Standard-Menüeinträge des Flash Players ausgeblendet werden sollen (Zeile 3). Im Konstruktor weisen Sie der Variable `cm` eine neue `ContextMenu`-Instanz zu und blenden gegebenenfalls über die `ContextMenu`-Funktion `hideBuiltInItems()` die Standard-Menüeinträge aus (Zeile 5). Bevor wir uns die `ContextMenu`-Klasse näher ansehen, ergänzen Sie noch folgende Codezeilen am Ende der Klasse, direkt vor der schließenden Klammer:

```
7: public function addBaseObject(bo: MovieClip){
8:   bo.menu = cm;
9: }
```

Diese Funktion erwartet als Parameter ein Objekt. Es handelt sich um dasjenige Element Ihrer Flash-Anwendung, das mit dem Kontextmenü versehen werden soll.

Schritt 4 Ein Zwischentest

Kontextmenü

Bei einem Test direkt in Flash wird ein geringfügig anderes Kontextmenü angezeigt als etwa später im Browser. Ebenso kann es in Abhängigkeit von der Flash Player-Version variieren. Das braucht Sie aber hier nicht zu kümmern, wichtig ist nur, dass Ihre individuellen Anpassungen dargestellt werden.

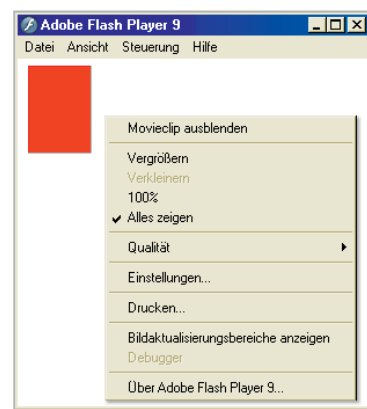
Abbildung 2 ►

Bei einem Rechtsklick auf den Hintergrund des Flash-Films erscheint das gewohnte Kontextmenü.

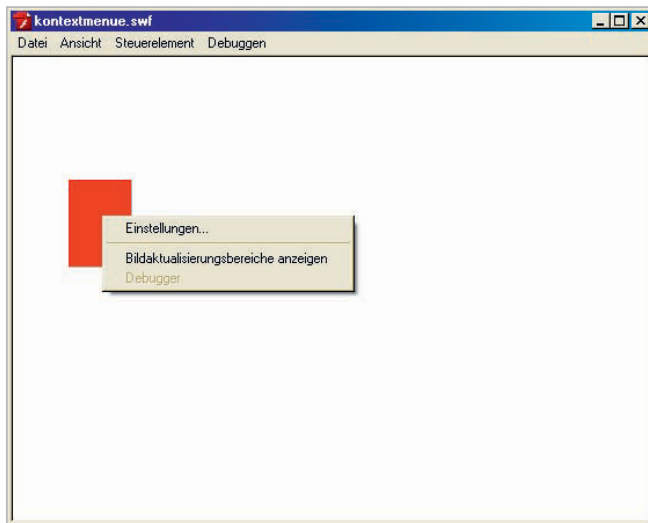
Sie können jetzt einen ersten Test machen: Wechseln Sie zur Flash-Datei, und öffnen Sie das ActionScript-Fenster. Geben Sie in Bild 1 der Hauptzeitleiste folgendes Skript ein:

```
var cm:CMenu = new CMenu(false);
cm.addBaseObject(mc);
```

Kompilieren Sie den Film mit `[Strg]/[F5] + [↵]`, und führen Sie einen Rechtsklick auf den Hintergrund aus – es erscheint das gewohnte Kontextmenü.



Führen Sie anschließend einen Rechtsklick auf den Movieclip »mc« aus – Sie sehen ein Kontextmenü, das nur noch über einige Standardeinträge verfügt.



◀ Abbildung 3

Wenn Sie mit der rechten Maustaste auf den Movieclip klicken, erscheint ein angepasstes Kontextmenü.

Einige Standard-Menüeinträge, die Sie in Abbildung 3 sehen, sind immer zu sehen und können nicht deaktiviert werden. Bei einem Rechtsklick im Browser werden nur noch die Elemente EINSTELLUNGEN und ÜBER ADOBE FLASH PLAYER 9 angeboten. Alle anderen Menüeinträge sind ausgeblendet, da Sie als Konstruktorparameter den Wert `false` angegeben haben und somit die Funktion `hideBuiltInItems()` ausgeführt wurde.

Die bisher programmierte Klasse `CMenu` bringt noch keinen wirklichen Vorteil gegenüber einer direkten Anwendung der `ContextMenu`-Klasse. Um die Vorteile der Klasse transparent zu machen, wollen wir einmal die »vorgefertigte« `ContextMenu`-Klasse unter die Lupe nehmen.

Mit der `ContextMenu`-Klasse kann ein Kontextmenü dynamisch zur Laufzeit erzeugt werden. Dieses kann dann entweder der Bühne, einem Movieclip, Button oder Textfeld zugeordnet werden. Eine Instanz der `ContextMenu`-Klasse verfügt über die Eigenschaft `customItems` – ein Array, das Kontextmenüeinträge enthält. Diese wiederum müssen in Form von speziellen Objekten vom Typ `ContextMenuItems` vorliegen.

Veranschaulichen Sie sich das anhand eines kleinen Beispiels: Erstellen Sie ein neues Flash-Dokument mit beliebigen Einstellungen. Platzieren Sie einen Movieclip auf der Bühne, und geben Sie ihm den Instanznamen »mc«. In Bild 1 der Hauptzeitleiste geben Sie folgendes Skript ein:

Schritt 5

Die `ContextMenu`-Klasse

```

var cm:ContextMenu = new ContextMenu();
var cm_eintraege:Array = new Array();
var cmi1:ContextMenuItem = new
ContextMenu("Eintrag 1", fkt1);
var cmi2:ContextMenuItem = new ContextMenuItem("Eintrag 2",
    fkt2);
cm_eintraege.push(cmi1, cmi2);
cm.customItems = cm_eintraege;
function fkt1(){
    trace("Eintrag 1 gewählt");
}
function fkt2(){
    trace("Eintrag 2 gewählt");
}
mc.menu = cm;

```

Hier wird ein Kontextmenü erstellt und dem Movieclip `mc` zugewiesen. Die Einträge müssen als separate Objekte manuell erstellt werden; ebenso bleibt das Erstellen und Verwalten des Arrays Ihnen überlassen. Eine Klasse kann Ihnen das ein für alle mal abnehmen.

Schritt 6 Menüeinträge anlegen

Die bisherige Klasse kann zwar ein Kontextmenü erstellen und einem beliebigen Objekt zuweisen, jedoch fehlt noch das Wichtigste – die Menüeinträge. Hierfür erstellen Sie eine öffentliche (`public`) Funktion. Ergänzen Sie im ActionScript-Dokument unterhalb der Funktion `addBaseObject()` folgenden Code:

```

1: public function addCMenuItem(itemName:String, menuText:String,
   callback:Function, separator:Boolean):Boolean{
2:   for(var i:Number=0; i<cm.customItems.length; i++){
3:     if((cm.customItems[i] == this["Item_"+itemName]) || (cm.
       customItems[i].caption == menuText)) return false;
4:   }
5:   this["Item_"+itemName] = new ContextMenuItem(menuText,
       callback);
6:   this["Item_"+itemName].separatorBefore = separator;
7:   cm.customItems.push(this["Item_"+itemName]);
8:   return true;
9: }

```

Da die Funktion als `public` deklariert wird, kann Sie außerhalb der Klasse verwendet werden. Als Parameter erwartet sie:

- ▶ einen Namen für das neue Menüelement
- ▶ einen Text, der dann so im Kontextmenü zu sehen ist
- ▶ eine Funktion, die bei Auswahl des Menüpunktes ausgeführt wird
- ▶ eine boolesche Variable, die angibt, ob oberhalb des Menüeintrags im Kontextmenü eine Trennlinie erscheinen soll. (So können Sie verwandte Menüeinträge optisch zu Blöcken zusammenfassen.)

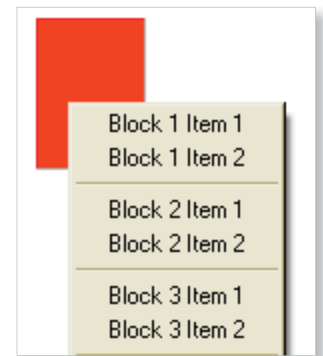
Ab Zeile 2 wird das `customItems`-Array des Kontextmenüs in einer `for`-Schleife durchlaufen. Beim ersten Aufruf hat dieses natürlich die Länge Null – somit wird der Anweisungsblock nicht ausgeführt. Befinden sich aber bereits Elemente im Array, wird für jedes Element geprüft, ob bereits ein Element mit demselben Namen existiert. In diesem Fall bricht die Funktion mit einer `return`-Anweisung ab (Zeile 3) und liefert `false` zurück. Dasselbe gilt für den unsinnigen Fall, dass ein zweiter Menüeintrag mit einem bereits vorhandenen Text erzeugt wird. Da solche Menüeinträge vom Benutzer nicht voneinander zu unterscheiden wären, wird das von der `ContextMenuItem`-Klasse erst gar nicht unterstützt, weshalb es auch an dieser Stelle abgefangen werden sollte.

Der so ermittelte Rückgabewert ermöglicht Ihnen, beim Anlegen neuer Menüeinträge zu überprüfen, ob das Element erfolgreich angelegt wurde. Sie können hierzu die Funktion wie im folgenden Beispielskript einfach wie eine boolesche Variable verwenden:

```
if(my_CMenu.addItem("option1", "Option1", callback)){
    trace("Menüelement angelegt");
}else{
    trace("Menüelement existiert bereits");
}
```

Hier wird die Funktion `addItem` für eine Instanz von `CMenu` mit dem Namen `my_CMenu` aufgerufen. Der Code bewirkt, dass ein Element mit dem Namen »Item_option1« angelegt wird, was durch die Ausgabe »Menüelement angelegt« bestätigt wird. Führen Sie aber exakt denselben Code danach erneut aus, wird kein weiteres Element angelegt, da es bereits existiert. Die Ausgabe lautet in diesem Fall »Menüelement existiert bereits«.

Wenn die Überprüfung ergeben hat, dass das Element noch nicht existiert, wird es in der nächsten Zeile angelegt. Dem so angelegten neuen Menüeintrag werden der darzustellende Text (`menuText`) sowie der Name einer Funktion (`callback`) übergeben. Diese wird ausgeführt, wenn der entsprechende Eintrag im Kontextmenü gewählt wird. Sie kann auch im Nachhinein über die Eigenschaft `onSelect` festgelegt werden, wie Sie in Schritt 9 sehen werden.



▲ **Abbildung 4**
Trennlinien im Kontextmenü

Der booleschen `ContextMenuItem`-Eigenschaft `separatorBefore` wird in Zeile 6 der übergebene Wert zugewiesen, um gegebenenfalls eine Trennlinie darzustellen.

Schritt 7

Menüeinträge entfernen

Menüeinträge sollen natürlich auch wieder entfernt werden können. Hierzu verwenden wir eine weitere Funktion. Ergänzen Sie folgenden Code unter der in Schritt 6 erstellten Funktion:

```
1: public function removeCMenuItem(itemName:String):Boolean{
2:   for(var i:Number=0; i<cm.customItems.length; i++){
3:     if(cm.customItems[i] == this["Item_"+itemName]){
4:       cm.customItems.splice(i, 1);
5:       return true;
6:     }
7:   }
8:   return false;
9: }
```

Wie beim Anlegen von Menüeinträgen, wird auch hier das `customItems`-Array komplett durchlaufen. Als Parameter wird der beim Anlegen verwendete Name erwartet. Wird eine Übereinstimmung festgestellt, wird der Eintrag über die `splice`-Methode aus dem Array entfernt (Zeile 4). Je nachdem, ob das Element erfolgreich entfernt wurde oder nicht, wird `true` oder `false` zurückgegeben.

Schritt 8

Menüpunkte an- und abschalten

Um kontrollieren zu können, wann ein Menüeintrag aktiv ist, fehlt noch eine entsprechende Funktion. Diese ist sehr einfach zu realisieren – ergänzen Sie folgenden Code, wieder unter der letzten Funktion:

```
public function enable(itemName:String, isEnabled:Boolean):Void{
    this["Item_"+itemName].enabled = isEnabled;
}
```

Die Funktion erwartet als Parameter den Namen des Menüpunkts sowie eine boolesche Angabe, ob dieser aktiv oder inaktiv gesetzt werden soll. Dem entsprechenden `ContextMenu`-Objekt wird ganz einfach über die `enabled`-Eigenschaft der übergebene Wert zugeordnet.

Sie haben jetzt eine Klasse, die es Ihnen ermöglicht, auf einfache Art und Weise Kontextmenüs zu erstellen und zu verwalten. Um sie in Aktion zu sehen, ändern Sie in der fla-Datei das Skript auf Bild 1 folgendermaßen:

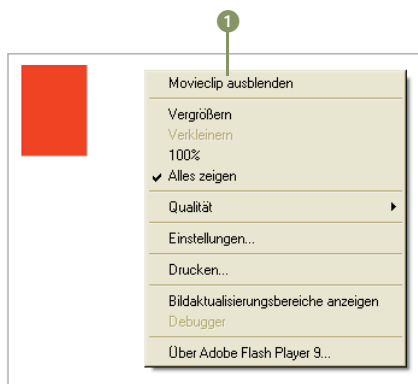
```
var cm:CMenu = new CMenu(false);
cm.addCMenuItem("rot", "Movieclip rot färben", rot);
cm.addCMenuItem("blau", "Movieclip blau färben", blau);
cm.addBaseObject(mc);
function rot(){
    setNewColor(mc, 0xff0000);
}
function blau(){
    setNewColor(mc, 0x0000ff);
}
function setNewColor(mc:MovieClip, 12: myColor:Number){
    var newColor:Color = new Color(mc);
    newColor.setRGB(myColor);
}
```

Im nächsten Schritt wollen wir die Klasse um ein nützliches Feature erweitern.

Erweiterungen von Klassen sind in jede Richtung denkbar. Hier erweitern Sie die Kontextmenü-Klasse exemplarisch um die Möglichkeit, einen Menüpunkt als Alternative zu einem anderen Menüpunkt zu definieren. Stellen Sie sich beispielsweise vor, Sie wollen über das Kontextmenü einen Movieclip ein- oder ausblenden. Die jeweils anzubietende Option ist abhängig davon, ob der Movieclip gerade sichtbar ist oder nicht.

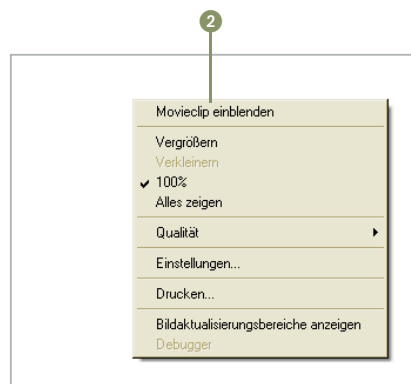
Schritt 9

Erweitern der Kern-funktionalität



▲ **Abbildung 5**

Ist der Movieclip sichtbar, soll eine Ausblenden-Option angeboten werden: MOVIECLIP AUSBLENDEN ①



▲ **Abbildung 6**

Nachdem der Movieclip ausgeblendet wurde, soll er auch wieder eingeblendet werden können: MOVIECLIP EINBLENDEN ②

Die beiden Optionen sollen immer nur alternativ angeboten werden. Die entsprechende Funktion benötigt zwei Menüeinträge, die auch vorhanden sein müssen. Ergänzen unter der zuletzt eingefügten Funktion folgende Zeilen:

```
1: public function setAlt(firstItem:String, secondItem:String,
   hideAlt:Boolean):Boolean{
2:   var firstItemExists:Boolean = false;
3:   var secondItemExists:Boolean = false;
4:   for(var i:Number=0; i<cm.customItems.length; i++){
5:     if(cm.customItems[i] == this["Item_"+firstItem])
       firstItemExists = true;
6:     if(cm.customItems[i] == this["Item_"+secondItem])
       secondItemExists = true;
7:   }
8:   if(!(firstItemExists&&secondItemExists)||firstItem==
       secondItem) return false;
9:   }
```

In den Zeilen 2 und 3 werden lokale boolesche Variablen mit dem Wert `false` erzeugt. Anschließend wird das Array der Menüeinträge durchlaufen, und jede der Variablen auf `true` gesetzt, wenn eine Übereinstimmung gefunden wurde. In Zeile 8 wird die Funktion mit dem Rückgabewert `false` abgebrochen, wenn einer der beiden Menüeinträge nicht existiert oder beide gleich sind (was unsinnig wäre). Ansonsten wird die eigentliche Funktion ausgeführt. Der Code muss direkt unter Zeile 8 eingefügt werden:

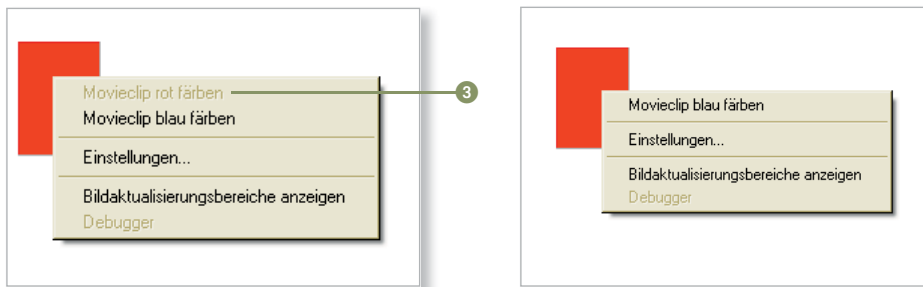
```
9:   hideAlt?this["Item_"+secondItem].visible =
       false:this["Item_"+secondItem].enabled = false;
10:  var oldCallback1:Function = this["Item_"+firstItem].
    onSelect;
11:  var oldCallback2:Function = this["Item_"+secondItem].
    onSelect;
12:  var scope:Object = this;
13:  this["Item_"+firstItem].onSelect = function(){
14:    hideAlt?scope["Item_"+firstItem].visible =
       false:scope["Item_"+firstItem].enabled = false;
15:    hideAlt?scope["Item_"+secondItem].visible =
       true:scope["Item_"+secondItem].enabled = true;
16:    oldCallback1();
17:  }
18:  this["Item_"+secondItem].onSelect = function(){
```

```

19: hideAlt?scope["Item_"+firstItem].visible =
    true:scope["Item_"+firstItem].enabled = true;
20: hideAlt?scope["Item_"+secondItem].visible =
    false:scope["Item_"+secondItem].enabled = false;
21: oldCallback2();
22: }
23: return true;

```

In Zeile 9 wird der boolesche Parameter `hideAlt` ausgewertet. Dieser gibt an, ob die jeweils inaktive Option ausgegraut oder gar nicht sichtbar sein soll. Je nachdem, wird diese entweder über `visible` oder `enabled` inaktiv gesetzt.



◀◀ **Abbildung 7**
Sie können die inaktive Option ausgrauen ③ ...

◀ **Abbildung 8**
... oder ganz ausblenden.

In den Zeilen 10 und 11 werden die an die Menüeinträge gekoppelten Funktionen (`onSelect`) zwischengespeichert. Schließlich werden die `onSelect`-Funktionen neu definiert. Hierbei wird die jeweils als Alternative angegebene Option in Abhängigkeit von dem Parameter `hideAlt` entweder ausgegraut oder ausgeblendet (Zeilen 14 und 15 sowie 19 und 20) und dann die zwischengespeicherte, alte `onSelect`-Funktion ausgeführt. So wird die ursprüngliche Funktionalität nicht beeinträchtigt. Zum Schluss wird die erfolgreiche Ausführung der Methode durch die Rückgabe von `true` bestätigt.

Seit dem letzten Zwischentest hat sich viel getan. Wechseln Sie daher in das Flash-Dokument, um die eben erstellte Klasse exemplarisch mit voller Funktionalität anzuwenden. Erweitern Sie das Skript in Bild 1 der Hauptzeitleiste folgendermaßen:

```

1: var cm:CMenu = new CMenu(false);
2: cm.addCMenuItem("rot", "Movieclip rot färben", rot);
3: cm.addCMenuItem("blau", "Movieclip blau färben", blau);
4: cm.addBaseObject(mc);

```

Schritt 10

Anwenden der Klasse

```

5:  function rot(){
6:      setNewColor(mc, 0xff0000);
7:  }
8:  function blau(){
9:      setNewColor(mc, 0x0000ff);
10: }
11: function setNewColor(mc:MovieClip, myColor:Number){
12:     var newColor:Color = new Color(mc);
13:     newColor.setRGB(myColor);
14: }
15: cm.setAlt("blau", "rot", false);
16: var cm2:CMenu = new CMenu(true);
17: cm2.addCMenuItem("ausblenden", "Movieclip ausblenden",
    function(){mc._visible=false;});
18: cm2.addCMenuItem("einblenden", "Movieclip einblenden",
    function(){mc._visible=true;});
19: cm2.addBaseObject(_root);
20: cm2.setAlt("ausblenden", "einblenden", true)

```

In Zeile 1 wird eine Instanz der CMenu-Klasse erstellt. In den Zeilen 2 und 3 werden Menüeinträge definiert, die als letzten Parameter einen Funktionsnamen erwarten. Diese Funktionen werden in den Zeilen 5 bis 10 definiert, nachdem in Zeile 4 der Movieclip `mc` als Basis-Objekt festgelegt wurde.

Sie färben unter Zuhilfenahme einer weiteren Funktion (`setNewColor`, Zeilen 11 bis 15) den Movieclip rot bzw. blau ein.

In Zeile 15 kommt die `setAlt`-Methode zum Einsatz. Dieselbe Prozedur wird dann noch einmal durchlaufen, diesmal für das Kontextmenü der Bühne und mit dem Ein- bzw. Ausblenden des Movieclips als Funktionalität.

Abbildung 9 ►

Der Movieclip kann über das Kontextmenü blau gefärbt werden.

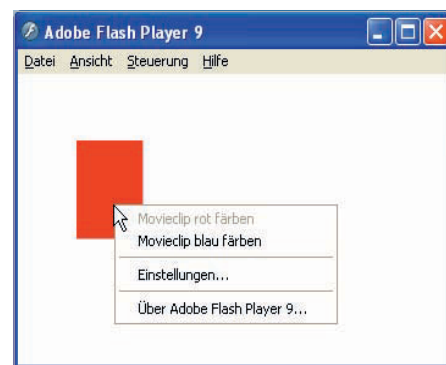


Abbildung 10 ►►

Ebenso führt auch der Weg zurück zum roten Movieclip.



Laufband

Ein Bilder-Laufband mit der Maus steuern

Ziel dieses Workshops ist, ein Laufband aus Bildern zu programmieren, das scheinbar unendlich nach links und rechts gescrollt werden kann. Die Maus wird dabei die Laufband-geschwindigkeit steuern. Das Ganze werden Sie in einer wieder verwendbaren Klasse realisieren, die Sie später in Ihre eigenen Projekte einbinden können.

Zielsetzungen:

Bilder importieren

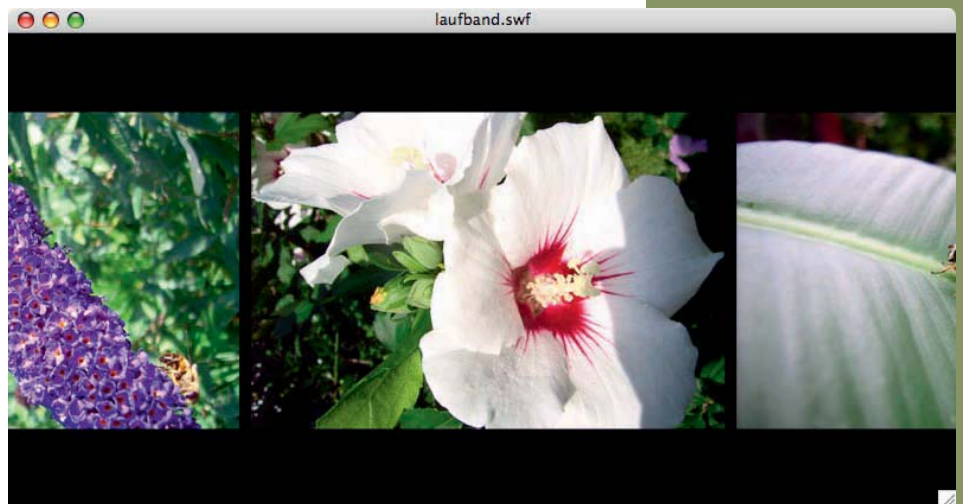
Bilder in einer Reihe positionieren

Laufband-Engine programmieren

[Ergebnis: 26_LAUFBAND •

LAUFBAND.SWF]

```
class Ticker {
    private var target : MovieClip;
    private var ticker : MovieClip;
    private var items : Array;
    private var width : Number;
    private var margin : Number;
    private var totalWidth : Number;
    public function Ticker(target : MovieClip, linkages:Array,
        width : Number, margin : Number)
    {
        this.target = target;
        this.width = width || 540;
        this.margin = margin || 5;
        ticker = target.createEmptyMovieClip("ticker", 1);
        attachItems(linkages);
        ticker.onEnterFrame = mx.utils.Delegate.create(this,
            onEnterFrame);
    }
    private function attachItems(linkages : Array) : Void
    {
        items = new Array();
```



Schritt 1 Vorbereitungen

Im ersten Schritt legen Sie einen neuen Ordner auf Ihrer Festplatte an und nennen ihn LAUFBAND. Hier werden Sie die von Ihnen erstellten Beispiele zu diesem Workshop abspeichern.

Schritt 2 Flash-Datei erstellen

Erstellen Sie ein neues AS2-Flash-Dokument mit der Grösse 800x400 Pixel und einer BILDRATE von 120bps. Die hohe Bildrate wird die Animationen flüssiger machen. Stellen Sie die HINTERGRUNDFARBE auf Schwarz. Speichern Sie das Dokument unter dem Namen LAUFBAND.FLA in dem Projektordner.

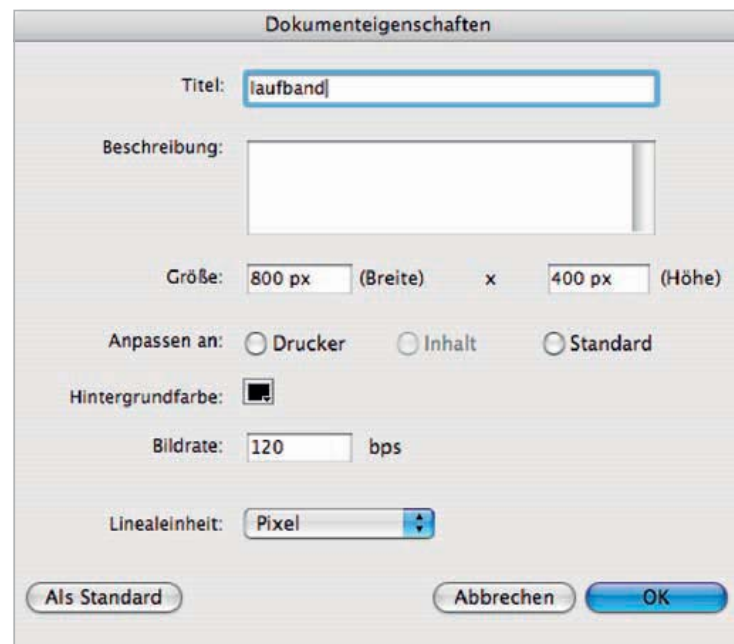
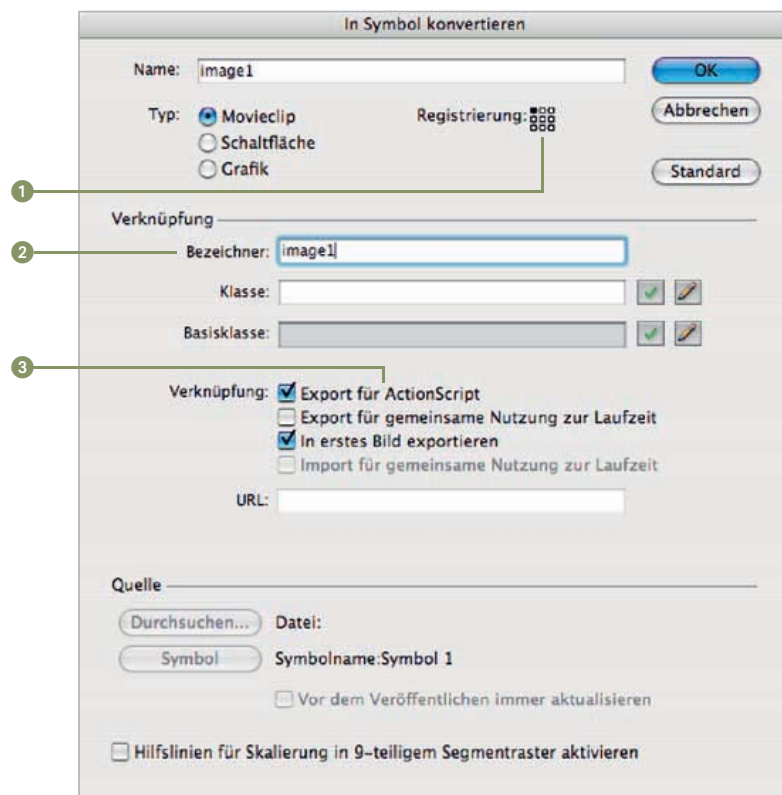


Abbildung 1 ►
Dokumenteigenschaften für
das Laufband einstellen

Schritt 3 Flash-Film vorbereiten

Importieren Sie über DATEI • IMPORTIEREN • IN BÜHNE IMPORTIEREN alle Bilder aus dem Projektordner 26_LAUFBAND • IMAGES von der Buch-DVD auf die Bühne. Sie können bei der Dateiauswahl gleich alle Dateien mit gedrückter -Taste auswählen. Die Bilder befinden sich nun auf der Bühne. Markieren Sie eins nach dem anderen, und erstellen Sie jeweils mit  einen Movieclip. Benennen Sie die Movieclips fortlaufend mit »image1« bis »image4«. Aktivieren Sie den EXPORT FÜR ACTIONSCRIPT . Wenn Sie zuvor den Namen eingegeben haben, entspricht automatisch auch der BEZEICHNER  diesem Namen. Achten Sie darauf, dass sich der Registrierungspunkt in der linken oberen Ecke  befindet. Bestätigen Sie Ihre Eingabe.



◀ **Abbildung 2**

Das erste Bild wird in einen Movieclip umgewandelt.

Wenn Sie den ersten Movieclip erzeugt haben, löschen Sie diesen wieder von der Bühne, damit die darunterliegenden Bilder nicht überdeckt sind. Die Movieclips befinden sich nun in der Bibliothek.

Wählen Sie **DATEI • NEU • ACTIONSCRIPT-DATEI**. Speichern Sie die Datei unter dem Namen **TICKER.AS** im Projektordner. In dieser Klasse werden Sie das Laufband programmieren.

Das Prinzip des Laufbands ist ganz einfach. Als Elemente des Laufbands dienen die Movieclips aus der Bibliothek. Zuerst positionieren Sie die Elemente nebeneinander. Durch den Abstand der Maus zur Mitte des Laufbands errechnen Sie einen Wert, mit der Sie die x-Position von jedem Element anpassen. Die Elemente fahren dann entweder nach links oder nach rechts. Wenn ein Element aus dem sichtbaren Bereich hinausfährt, reihen Sie dieses wieder an dem entgegengesetzten Ende des Laufbands ein. Dadurch entsteht eine Endlosschleife und Sie können das Laufband unendlich in beide Richtungen bewegen.

Schritt 4

ActionScript-Datei erstellen

Schritt 5

Das Laufband-Prinzip

Schritt 6

Klassengerüst programmieren

Wechseln Sie zur Datei TICKER.AS und programmieren Sie nun das Klassengerüst mit Konstruktor:

```
1: class Ticker {  
2:     public function Ticker()  
3:     {  
4:     }  
5: }
```

Schritt 7

Variablen deklarieren

Um das Laufband programmieren zu können, müssen verschiedene Klassenvariablen deklariert werden, auf die Sie aus jeder Methode Ihrer Klasse zugreifen können:

- ▶ Als Erstes benötigen Sie einen Movieclip, auf dem die Elemente des Laufbands erzeugt werden.
- ▶ Des Weiteren benötigen Sie ein Array, in dem Sie die Elemente abspeichern. Ein Array bietet den Vorteil, dass man darin die Elemente in der Reihenfolge abspeichern kann, wie sie auf der Bühne aufgereiht sind. Dadurch weiß man immer, wo sich welches Element befindet, und kann dieses, wenn es aus dem Bild fährt, an das andere Ende der Reihe anhängen und das Array dementsprechend anpassen.
- ▶ Sie benötigen außerdem eine Variable, die den Abstand zwischen den Elementen festlegt.
- ▶ Die Breite aller Elemente zusammen muss ebenfalls gespeichert werden. Die Gesamtbreite ist hilfreich, um ein Element am Ende des Laufbands zu positionieren.
- ▶ Außerdem soll in einer Variablen festgelegt werden, wie lang bzw. breit das Laufband selbst sein soll, damit dieses nicht zwangsläufig die ganze Bühne beansprucht.

Deklarieren Sie daher folgende Klassenvariablen über dem Konstruktor:

```
1: private var target : MovieClip;  
2: private var ticker : MovieClip;  
3: private var items : Array;  
4: private var width : Number;  
5: private var margin : Number;  
6: private var totalWidth : Number;
```

In Zeile 1 deklarieren Sie `target` vom Typ `MovieClip`. In `target` soll später ein weiterer Movieclip erzeugt werden, der durch die Klassenvariable `ticker` referenziert wird. In `ticker` werden die Elemente des Laufbands geladen. In Zeile 3 deklarieren Sie `items`, das Array, in dem die Elemente gespeichert werden. Die Variable `width` wird die Breite des Laufbands bestimmen. In Zeile 5 deklarieren Sie `margin`, den Abstand zwischen den Elementen. `totalWidth` wird die gesamte Breite – aller Elemente zusammen gerechnet – abspeichern.

Das Laufband soll später ganz einfach durch den Aufruf des Konstruktors initialisiert und gestartet werden. Erweitern Sie zu diesem Zweck den Konstruktor folgendermaßen:

```
1: public function Ticker(target : MovieClip, linkages:Array,  
    width : Number, margin : Number)  
2: {  
3:     this.target = target;  
4:     this.width = width || 540;  
5:     this.margin = margin || 5;  
6:     ticker = target.createEmptyMovieClip("ticker", 1);  
7:     attachItems(linkages);  
8: }
```

Der Konstruktor der Ticker-Klasse erwartet vier Parameter: Der erste Parameter `target` ist der Movieclip, auf dem die Laufleiste erzeugt wird. In Zeile 3 speichern Sie diesen Parameter in der Klassenvariablen `target`. Bei dem zweiten Parameter `linkages` handelt es sich um ein Array, das die Verknüpfungsnamen der Movieclips beinhaltet, die aus der Bibliothek in das Laufband aufgenommen werden.

Der nächste Parameter `width` wird der Klassenvariablen `width` zugeordnet und damit die Breite des Laufbands festgelegt. Sollte dieser Parameter dem Konstruktor nicht übergeben worden sein, weisen Sie `width` in Zeile 4 automatisch den Wert 540 zu. In Zeile 5 wird der Klassenvariablen `margin` der Parameter `margin` zugeordnet. Sollte dieser Wert auch nicht gesetzt sein, setzen Sie den Abstand der Elemente auf 5 Pixel. In der nächsten Zeile erzeugen Sie nun einen Movieclip innerhalb von `target` und speichern diesen in der Klassenvariablen `ticker`. Eine Zeile weiter rufen Sie eine Methode `attachItems()` auf und übergeben dieser den `linkages`-Array. `attachItems()` soll die Movieclips auf die Bühne bringen. Die Methode werden Sie im nächsten Schritt programmieren.

Schritt 8

Variablen initialisieren

Verknüpfungsname nachträglich ergänzen

Haben Sie vergessen, beim Anlegen des Movieclips einen Bezeichner anzugeben, können Sie das auch nachträglich noch tun. Markieren Sie dazu das Element in der Bibliothek und klicken Sie auf das i-Symbol ganz unten. Flash öffnet den Dialog **IN SYMBOL KONVERTIEREN**. Dort können Sie den **EXPORT FÜR ACTIONSCRIPT** aktivieren und anschließend den **BEZEICHNER** vergeben.

Schritt 9

Movieclips auf der Bühne erstellen

In diesem Schritt werden Sie die Movieclips aus der Bibliothek auf der Bühne instantiiert und in einer Reihe anordnen. Schreiben Sie die Methode `attachItems()` folgendermaßen direkt unter den Konstruktor:

```
1: private function attachItems(linkages : Array) : Void
2: {
3:     items = new Array();
4:     totalWidth = 0;
6:     for(var i:Number = 0; i< linkages.length; i++){
6:         var item : MovieClip = ticker.attachMovie(linkages[i],
            'item_'+i, i);
7:         item._x = totalWidth;
8:         totalWidth += item._width + margin;
9:         items.push(item);
10:    }
11: }
```

Die Methode erwartet als Parameter das Array, das die Verknüpfungsnamen zu den Movieclips in der Bibliothek enthält. Bevor Sie diese auf die Bühne holen, weisen Sie in Zeile 3 der Klassenvariablen `items` ein neues Array zu. `items` soll die erzeugten Movieclips speichern. Der Klassenvariablen `totalWidth` weisen Sie zuvor in Zeile 4 den Wert 0 zu, da bis jetzt noch keine Movieclips erzeugt wurden.

In der nächsten Zeile durchlaufen Sie `linkages` mit einer `for`-Schleife. Mit Hilfe von `attachMovie()` und dem in `linkages` gespeicherten Verknüpfungsnamen erzeugen Sie innerhalb von `ticker` Instanzen der Movieclips aus der Bibliothek. Den Movieclips weisen Sie über den zweiten Parameter von `attachMovie`, `'item_'+i`, dynamisch einen Namen zu, bestehend aus dem Präfix »item_« und dem Wert von `i`. Damit Sie den neu erzeugten Movieclip allerdings nicht über diese Zusammensetzung ansprechen müssen, speichern Sie ihn temporär in der Variablen `item`.

In Zeile 7 weisen Sie der `_x`-Eigenschaft von `item` den Wert von `totalWidth` zu. Da diese Variable anfänglich den Wert 0 hat, wird das erste Element somit auf x-Position 0 erstellt. Daraufhin erhöhen Sie `totalWidth` um die Breite von `item` plus den Wert von `margin`. Da Sie zu Beginn des Workshops in allen Movieclips den Registrierungspunkt in der oberen linken Ecke angelegt haben, müssen Sie die Breite des Movieclips mit einberechnen, um die Position des nächsten `item` zu bestimmen.

In Zeile 9 fügen Sie `item` dem `items`-Array hinzu. Die Elemente befinden sich nun entsprechend ihrer Reihenfolge auf der Bühne im `items`-Array.

Wechseln Sie nun zur Hauptzeitleiste. Klicken Sie auf das erste Bild der ersten Ebene, und öffnen Sie mit **[F9]** oder **[Alt]+[F9]** am Mac das Action-Script-Fenster. Geben Sie folgenden Code ein:

```
1: var items:Array = new Array("image1",  
    "image2","image3","image4");  
2: var holder:MovieClip = createEmptyMovieClip("holder",1);  
3: var ticker:Ticker = new Ticker(holder,items,600,10);  
4: holder._y = Stage.height/2-holder._height/2;
```

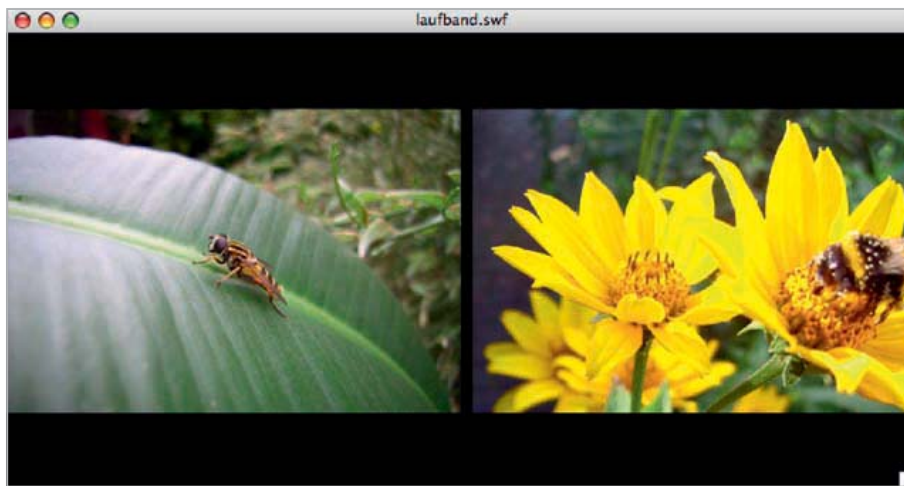
In der ersten Zeile erzeugen Sie ein Array `items`, in dem Sie die Verknüpfungsnamen der in der Bibliothek befindlichen Movieclips in Form von Strings abspeichern.

In Zeile 2 erzeugen Sie einen neuen Movieclip `holder`. Auf diesem soll das Laufband erzeugt werden.

In der nächsten Zeile erzeugen Sie nun eine Instanz der Klasse `Ticker` und übergeben dieser die notwendigen Parameter. Der dritte Parameter soll die Breite des Laufbands auf 600 Pixel setzen. Dieser Wert wird allerdings zum momentanen Stand keine Wirkung zeigen, da Sie die Breitenbeschränkung erst in den nächsten Schritten realisieren werden.

Mit der letzten Zeile positionieren Sie `holder` vertikal zentriert.

Testen Sie nun Ihren Flash-Film mit **[Strg]/[⌘]+[↵]**. Wie Sie sehen, befinden sich die Movieclips aus der Bibliothek nebeneinander gereiht auf der Bühne. Momentan sehen Sie allerdings nur zwei der Movieclips, da sich die weiteren jenseits des rechten Bühnenrandes befinden.



Schritt 10

Flash-Datei vorbereiten

Schritt 11

Film testen

◀ **Abbildung 3**
Die Bilder werden nebeneinander auf der Bühne platziert.

Schritt 12

Maske erstellen

Die Laufband-Klasse `Ticker` soll die Möglichkeit bieten, die Breite des Laufbands zu bestimmen. Stellen Sie sich vor, der Flash-Film, in den Sie das Laufband einbauen, hat eine Breite von 800 Pixel. Das Laufband soll allerdings nur 600 Pixel breit sein, weil sich daneben zum Beispiel noch ein Textfeld mit Beschreibungen der Bilder befindet. Wenn Sie das Laufband erzeugen, würden die Elemente jedoch über die gesamte Breite der Bühne verteilt werden, da diese ja ohne Beschränkung der Reihe nach positioniert werden.

Um dies zu verhindern, werden Sie eine Maske über den `ticker-Movieclip` legen. Wechseln Sie Sie zu diesem Zweck also wieder in die `Ticker`-Klasse, und schreiben Sie eine Methode `createMask()` unter die Methode `attachItems()` aus Schritt 9:

```
1: private function createMask() : Void
2: {
3:     var mask:MovieClip = target.
        createEmptyMovieClip('mask',2);
4:     mask.beginFill(0xffffffff);
5:     mask.moveTo(0,0);
6:     mask.lineTo(width,0);
7:     mask.lineTo(width,ticker._height);
8:     mask.lineTo(0,ticker._height);
9:     mask.lineTo(0,0);
10:    mask.endFill();
11:    ticker.setMask(mask);
12: }
```

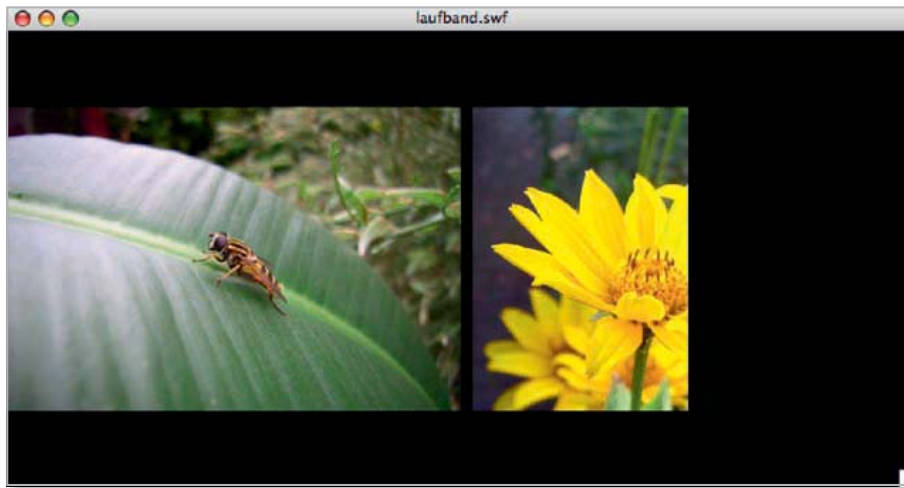
In Zeile 3 erzeugen Sie auf Ebene 2 von `target` einen neuen `Movieclip` `mask`. Dieser `Movieclip` befindet sich damit eine Ebene über dem `ticker-Movieclip`. Jetzt müssen Sie `mask` dynamisch mit einer Form füllen, welche der gespeicherten Breite `width` und der Höhe von `ticker` entspricht, um diesen damit maskieren zu können. Sie verwenden dafür die Zeichnen-Methoden von Flash. In den Zeilen 4 bis 10 zeichnen Sie ein Rechteck, das der Höhe von `ticker` und der Breite `width` entspricht. In Zeile 11 maskieren Sie den `ticker-Movieclip` mit `mask`.

Damit die Methode auch ihre Wirkung zeigt, muss sie noch aufgerufen werden. Fügen Sie dazu in die letzte Zeile der `attachItems`-Methode nach der `for`-Schleife die folgende Zeile ein:

```
createMask();
```

Jetzt wird nur noch der angegebene Bereich zu sehen sein.

Testen Sie nun wieder Ihren Flash-Film mit `Strg/⌘ + ⬅`. Wie Sie sehen, ist das Laufband nun nach genau 600 Pixeln abgeschnitten. Ihre Maskierung funktioniert.



Schritt 13

Film testen

◀ **Abbildung 4**
Das Laufband wird maskiert.

Nun geht es in den Endspurt. Was noch fehlt, ist die Engine, die die Movieclips unendlich in beide Richtungen fahren lässt. Programmieren Sie dafür eine Methode `onEnterFrame()`, die später in jedem Frame aufgerufen wird. Die Methode wird ans Ende der Klasse, unter die zuletzt eingefügte Methode gesetzt:

```
1: private function onEnterFrame() : Void
2: {
3:     var speed : Number = int((width/2-target._xmouse)/30);
4:     for(var i:Number = 0; i< items.length; i++){
5:         items[i]._x += speed;
6:     }
7:     var firstitem:MovieClip = items[0];
8:     var lastitem:MovieClip = items[items.length-1];
9:     if(speed < 0){
10:        if(firstitem._x < -firstitem._width){
11:            firstitem._x = lastitem._x+lastitem._width+margin;
12:            var item:Object = items.shift();
13:            items.push(item);
14:        }
15:    }
16:    else if(speed > 0){
17:        if(lastitem._x > width+lastitem._width){
18:            lastitem._x = firstitem._x-margin-lastitem._width;
```

Schritt 14

Movieclips am laufenden Band

```

19:         var item: Object = items.pop();
20:         items.unshift(item);
21:     }
22: }
23: }

```

In Zeile 3 erzeugen Sie eine Variable `speed`. Diese wird die Geschwindigkeit des Laufbands bestimmen. Sie errechnet sich aus dem Abstand der Maus zur Mitte des Laufbands. In der Mitte hat `speed` den Wert 0. Nach links und rechts wird sich der Wert jeweils vergrößern oder verkleinern. Den errechneten Wert teilen Sie noch durch 30, da das Ergebnis zu groß wäre und das Laufband zu schnell rollen würde.

Von Zeile 4 bis 6 durchlaufen Sie in einer `for`-Schleife das `items`-Array und addieren den `speed`-Wert auf die `_x`-Eigenschaft der darin befindlichen Movieclips.

In Zeile 7 speichern Sie eine Referenz auf das erste Element von `items` in der Variable `firstitem`. In der nächsten Zeile speichern Sie eine Referenz auf das letzte Element in `lastitem`. Diese beiden Referenzen benötigen Sie, um zu prüfen, ob das jeweilige Element über die Begrenzungen der Laufleiste hinausgefahren ist – um es dann neu zu positionieren. Und genau das programmieren Sie in den darauffolgenden Zeilen.

In Zeile 9 fragen Sie ab, ob `speed` einen negativen Wert enthält. Ist dies der Fall, bewegen sich die Elemente nach links. Deswegen fragen Sie in Zeile 10 ab, ob `firstitem` eine x-Position kleiner als die eigene Breite hat. Ist dies der Fall, befindet sich `firstitem` links außerhalb des sichtbaren Bereichs und Sie setzen die x-Position auf die Position von `lastitem` plus dessen Breite und `margin`, also an das rechte Ende der Reihe. Weil sich die Reihenfolge der Elemente geändert hat, müssen Sie die Sortierung des Arrays verändern. In Zeile 12 entfernen Sie das erste Element von `items`, zwischenspeichern dieses in der Variablen `item` und fügen es am Ende des Arrays wieder an.

In Zeile 16 beginnt das ganze Spiel von vorn, nur dass Sie nun abfragen, ob `speed` größer 0 ist und sich damit die Elemente nach rechts bewegen. In diesem Fall fragen Sie ab, ob die x-Position von `lastitem` größer als die in `width` festgelegte Breite des Laufbands plus der Breite von `lastitem` ist. Es würde sich also rechts außerhalb des sichtbaren Bereichs befinden. Für diesen Fall setzen Sie die x-Position von `lastitem` an den Anfang der Reihe. Nun müssen Sie wieder das Array anpassen. In Zeile 19 entfernen Sie das letzte Element von `items`, woraufhin Sie es an dessen erster Stelle wieder einsetzen. Das ist der ganze Zauber eines unendlichen Laufbands. Die Engine ist nun fertig.

Da die Laufband-Klasse nicht von `MovieClip` erbt und auch mit keinem `Movieclip` aus der Bibliothek verknüpft ist, wird die soeben programmierte `onEnterFrame`-Methode nicht von selbst aufgerufen. Deswegen weisen Sie nun dem `ticker-Movieclip` die `onEnterFrame`-Methode zu. Erweitern Sie zu diesem Zweck den Konstruktor um folgende Codezeile:

```
ticker.onEnterFrame = mx.utils.Delegate.create(this,
    onEnterFrame);
```

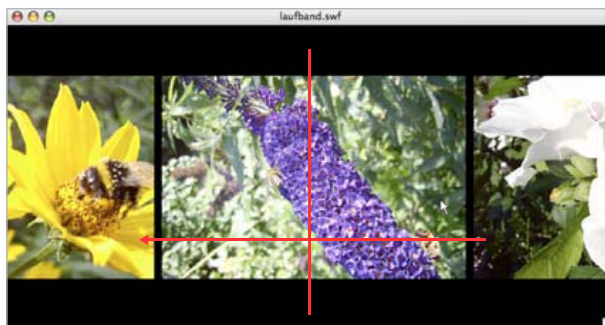
Sie verwenden hier die `Delegate`-Klasse, um in jedem Frame des `ticker-Movieclips` die `onEnterFrame`-Methode der `Ticker`-Klasse aufzurufen.

Bevor Sie den Film testen, wollen wir nun das Laufband wieder auf die volle Bühnenbreite bringen, damit das Ergebnis umso besser aussieht. Wechseln Sie dazu wieder zu dem Skript in der Hauptzeitleiste und passen Sie den Konstruktoraufruf so an, dass Sie als Laufbandbreite `Stage.width` übergeben, also die gesamte Bühnenbreite:

```
var ticker:Ticker = new Ticker(holder,items,Stage.width,10);
```

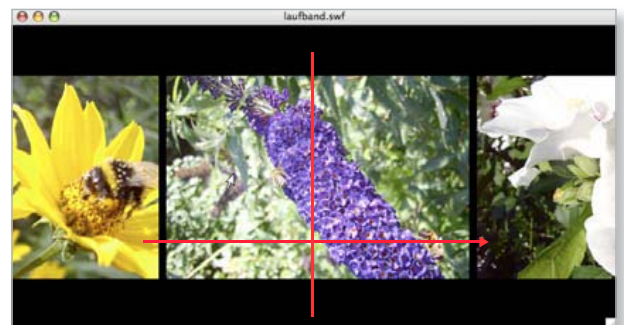
Testen Sie nun Ihren Flash-Film mit `[Strg]/[⌘]+[←]`. Bewegen Sie die Maus über die sich bewegende Laufleiste. Wie Sie sehen, fahren die Elemente nach links und rechts, je nach der Mausposition. Die Elemente tauchen immer wieder am anderen Ende auf.

Herzlichen Glückwunsch, Sie haben ein Laufband programmiert!



▲ **Abbildung 5**

Befindet sich der Mauszeiger rechts von der Mitte des Laufbands, bewegt es sich nach links.



▲ **Abbildung 6**

Befindet sich der Mauszeiger links von der Mitte des Laufbands, bewegt es sich nach rechts.

Schritt 15

`onEnterFrame` aufrufen

Schritt 16

Laufbandbreite maximieren

Schritt 17

Film testen

Slideshow

Bilder in einer Diashow präsentieren

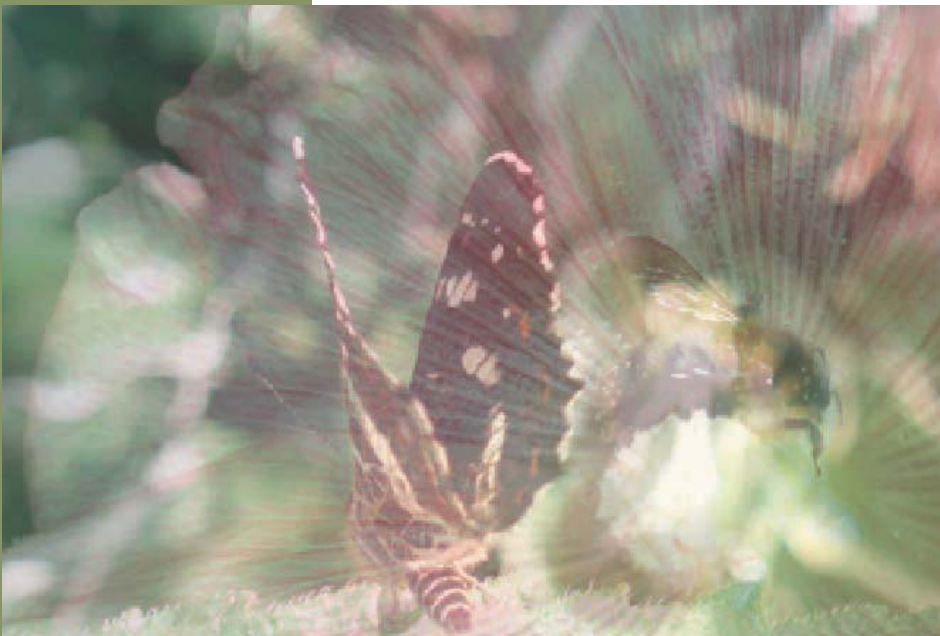
```
import mx.transitions.TransitionManager;
import mx.transitions.Transition;
import mx.transitions.Fade;
import mx.transitions.easing.Strong;
class SlideShow {
    private var clipLoader:ClipLoader;
    private var target:MovieClip;
    private var items:Array;
    private var currentItem:Number;
    private var numItemsLoaded:Number;
    private var seconds:Number;
    private var interval:Number;
    public function SlideShow(target:MovieClip, itemurls:Array,
    seconds:Number)
    {
        System.security.allowDomain('*');
        this.target = target;
```

Wollten Sie Ihren Bekannten schon immer Ihre Urlaubsbilder auf Ihrer Homepage präsentieren? Nun können Sie jeden einladen, sich vor dem Bildschirm entspannt zurückzulehnen, um Ihre Diashow zu bewundern! In diesem Workshop lernen Sie, wie Sie eine wieder verwendbare SlideShow-Klasse programmieren und wie Sie diese ganz einfach in Ihre Projekte einbinden können. Sie können externe Bilder oder Flash-Filme laden und diese nacheinander mit einer sanften Überblendung anzeigen.

Zielsetzungen:

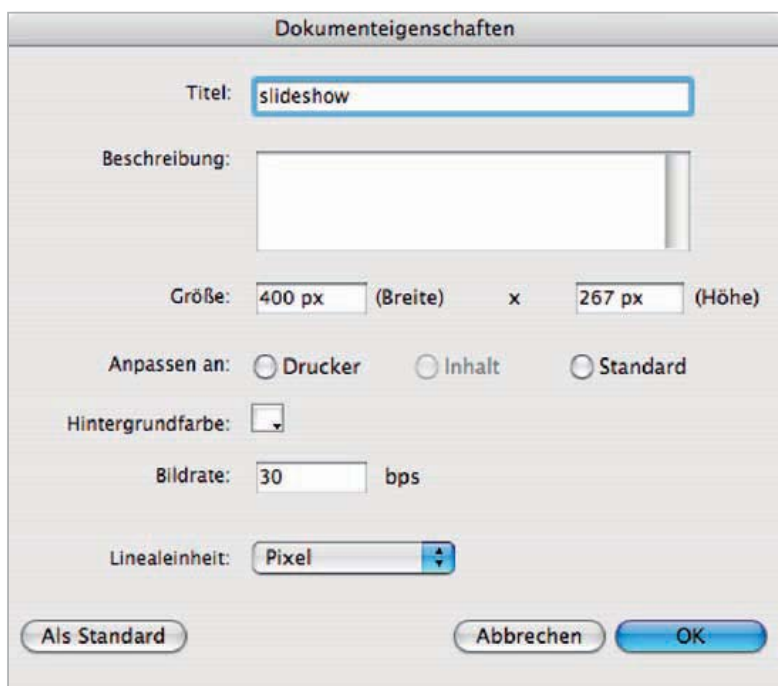
- Externe Bilder laden
- Bildüberblendung programmieren

[Ergebnis: 27_SLIDESHOW • SLIDESHOW.SWF]



Im ersten Schritt legen Sie einen neuen Ordner auf Ihrer Festplatte an und nennen ihn SLIDESHOW. Hier werden Sie die von Ihnen erstellten Dateien zu diesem Workshop abspeichern.

Erstellen Sie ein neues AS2-Flash-Dokument mit der Grösse 400x267 Pixel und einer BILDRATE von 30bps. HINTERGRUNDFARBE und sonstige Einstellungen spielen keine Rolle. Speichern Sie die Datei unter dem Namen SLIDESHOW.FLA im Projektordner.



Wählen Sie DATEI • NEU • ACTIONSCRIPT-DATEI. Speichern Sie die Datei unter dem Namen SLIDESHOW.AS im Projektordner.

Da eine Slideshow ohne Anzeige von Bildern wenig Sinn machen würde, werden Sie externe Bilder laden und anzeigen lassen. Zum Laden verwenden Sie die im Workshop »Cliploader« (21_CLIPLOADER) programmierte ClipLoader-Klasse. Diese wird das Laden vereinfachen. Kopieren Sie also die ActionScript-Datei CLIPLOADER.AS aus dem entsprechenden Projektordner oder von der Buch-DVD in Ihren SLIDESHOW-Ordner. Nun können Sie die Klasse ganz einfach verwenden.

Schritt 1 Vorbereitungen

Schritt 2 Flash-Datei erstellen

◀ **Abbildung 1**
Die Dokumenteigenschaften erreichen Sie über **[Strg]/[J]**.

Schritt 3 ActionScript-Datei erstellen

Schritt 4 ClipLoader kopieren

Schritt 5

Importe vornehmen

Die SlideShow-Klasse wird die geladenen Bilder nacheinander anzeigen. Sie werden dafür eine Überblendung entwickeln. Das aktuelle Bild wird ausgeblendet, während das nächste Bild eingeblendet wird. Diesen Effekt programmieren Sie mit der TransitionManager-Klasse.

Öffnen Sie die Datei SLIDESHOW.AS, und geben Sie folgende Import-Anweisungen ein:

```
1: import mx.transitions.TransitionManager;
2: import mx.transitions.Transition;
3: import mx.transitions.Fade;
4: import mx.transitions.easing.Strong;
```

In Zeile 1 importieren Sie die TransitionManager-Klasse. Mit dieser Klasse haben Sie eine Vielzahl von Möglichkeiten, Überblendungseffekte zu erstellen. Des Weiteren importieren Sie die Transition-Klasse. Mit Hilfe dieser Klasse werden Sie bestimmen, ob das Bild ein- oder ausgeblendet wird. In Zeile 3 importieren Sie die Fade-Klasse. Die Fade-Klasse bestimmt die Art der Überblendung. In diesem Fall handelt es sich um eine Alphakanal-Überblendung. Zu guter Letzt importieren Sie die Bewegungsklasse `Strong` aus dem `mx.transitions.easing`-Package. Diese wird die Eigenschaften der Animation der Überblendung bestimmen.

Adobe-Lifedocs

Informieren Sie sich in den Lifedocs von Adobe (<http://livedocs.adobe.com>) über die Möglichkeiten der TransitionManager-Klasse. Ihnen stehen noch weitere sehr interessante Überblendungen zur Verfügung, die Sie leicht in Ihre Slideshow einbauen können.

Schritt 6

Klassengerüst programmieren

Programmieren Sie nun den Klassenrumpf mit Konstruktor folgendermaßen und fügen Sie ihn direkt unter die letzte Codezeile aus Schritt 5 ein:

```
class SlideShow {
    public function SlideShow()
    {
    }
}
```

Schritt 7

Variablen deklarieren

Um die SlideShow-Klasse zu programmieren, müssen verschiedene Klassenvariablen deklariert werden, auf die Sie aus allen Methoden der Klasse zugreifen können. Nehmen Sie zu diesem Zweck folgende Deklarationen über dem Konstruktor vor:

```
1: private var clipLoader:ClipLoader;
```



```
2: private var target:MovieClip;
3: private var items:Array;
4: private var currentItem:Number;
5: private var numItemsLoaded:Number;
6: private var seconds:Number;
7: private var interval:Number;
```

Zuerst deklarieren Sie eine Variable `clipLoader`. In dieser werden Sie eine Instanz der `ClipLoader`-Klasse speichern.

Dann deklarieren Sie `target` vom Typ `MovieClip`. In diesen `Movieclip` werden Sie später die Bilder Ihrer Slideshow laden.

In Zeile 3 deklarieren Sie ein Array `items`. In `items` werden Sie Referenzen auf die `Movieclips` speichern, die erzeugt werden, um die Bilder darin zu laden.

Die in Zeile 4 deklarierte Variable `currentItem` wird eine Referenz auf die Position des aktuell angezeigten Bildes in dem Array `items` halten.

Danach deklarieren Sie `numItemsLoaded` vom Typ `Number`. Diese Variable dient zur Überprüfung, ob auch alle Bilder geladen wurden.

Die Variable `seconds` bestimmt die Anzahl an Sekunden, die ein Bild angezeigt bleibt, bevor es durch das nächste ausgetauscht wird.

In Zeile 7 deklarieren Sie eine numerische Variable `interval`, in der Sie ein Intervall speichern, das den Aufruf zum Überblenden der Bilder durchführen wird.

Der Konstruktor der `SlideShow`-Klasse wird drei Parameter erwarten:

- ▶ den `target-Movieclip`, auf dem die Bilder erzeugt werden
- ▶ ein Array, das URLs zu den zu ladenden Bildern enthält
- ▶ die Sekunden, wie lange ein Bild bis zur Überblendung angezeigt wird

Erweitern Sie den Konstruktor zu diesem Zweck unter der letzten Klassenvariablen aus Schritt 7 folgendermaßen:

```
1: public function SlideShow(target:MovieClip, itemurls:Array,
   seconds:Number)
2: {
3:     System.security.allowDomain('*');
4:     this.target = target;
5:     this.seconds = seconds;
6:     clipLoader = new ClipLoader();
```

Schritt 8

Anpassen des Konstruktors

Sicherheitseinstellungen

Das Sicherheitskonzept von Flash lässt nicht ohne weiteres zu, Dateien von einem entfernten System zu laden und auf deren Eigenschaften zuzugreifen. Die Security-Klasse bietet Möglichkeiten, diese Einstellungen zu beeinflussen. Sie können sich in der Flash-Hilfe näher über die Sicherheitseinstellungen informieren.

```
7:     clipLoader.addListener(this);
8:     loadItems(itemurls);
9: }
```

In Zeile 1 erweitern Sie den Konstruktor um die oben genannten Parameter. In Zeile 3 teilen Sie den Sicherheitseinstellungen von Flash mit, dass Sie jeder URL den Zugriff erlauben, damit der Überblendeffekt der TransitionManager-Klasse auf die geladenen Bilder zugreifen darf.

In Zeile 4 weisen Sie der Klassenvariablen `target` den übergebenen Parameter `target` zu. Dasselbe machen Sie mit `seconds`. In Zeile 6 instantiieren Sie die `clipLoader`-Variable mit einem Objekt der `ClipLoader`-Klasse. Daraufhin registrieren Sie `SlideShow` als Listener bei der `ClipLoader`-Instanz. In Zeile 8 rufen Sie eine Methode `loadItems()` auf und übergeben dieser das im Konstruktor erwartete Array `itemurls`, in dem die URLs zu den Bildern gespeichert sind. Diese Methode wird nun die Bilder laden.

Schritt 9 Bilder laden

Schreiben Sie nun die Methode `loadItems()` zum Laden der Bilder folgendermaßen unterhalb des Konstruktors:

```
1: private function loadItems(itemurls:Array):Void
2: {
3:     items = new Array();
4:     numItemsLoaded = 0;
5:     for(var i:Number = 0; i<itemurls.length; i++){
6:         var item:MovieClip = target.createEmptyMovieClip
           ('item'+i,i);
7:         items.push(item);
8:         clipLoader.loadClip(itemurls[i],item);
9:     }
10: }
```

Die Methode erwartet einen Parameter `itemurls` vom Typ `Array`. In diesem Array befinden sich die URLs zu den zu ladenden Bildern. Für jeden Eintrag des Arrays soll ein `Movieclip` erzeugt werden, in den das jeweilige Bild geladen wird. Bevor Sie allerdings das Laden einleiten, weisen Sie in Zeile 3 der Instanzvariablen `items` ein neues Array zu. In `items` werden die `Movieclips` gespeichert, in die die Bilder geladen werden. Der Klassenvariablen `numItemsLoaded` weisen Sie vor dem Laden den Wert 0 zu, da noch nichts geladen wurde.

Hiernach durchlaufen Sie in einer `for`-Schleife den Inhalt von `itemurls`. Für jede zu ladende Datei erzeugen Sie in Zeile 6 einen neuen `Movieclip`

und speichern diesen in `items`, um später durch die einfache Indexierung des Arrayinhalts darauf Zugriff zu haben.

Nun nutzen Sie in Zeile 8 die `loadClip`-Methode der `ClipLoader`-Klasse, um die externen Dateien in die soeben erzeugten Movieclips zu laden.

Wechseln Sie nun zur Hauptzeitleiste, und klicken Sie auf das erste Bild der ersten Ebene. Öffnen Sie mit `[F9]` bzw. `[Alt]+[F9]` am Mac das ActionScript-Fenster. Geben Sie folgenden Code ein:

```
1: var url:String = "http://www.flashfastforward.de/images/";
2: var items:Array = [url+"image1.jpg",url+"image2.jpg",
    url+"image3.jpg"];
```

Sparen Sie sich gleich etwas Tipparbeit: Da die Bilddateien von einem externen Server geladen werden, erzeugen Sie eine Variable `url` vom Typ `String`, in der Sie die URL zu dem Bilder-Server speichern. In der nächsten Zeile füllen Sie ein Array `items` mit den vervollständigten URLs zu den Bildern. Als Nächstes erzeugen Sie einen Movieclip, auf dem die Slideshow zum Einsatz kommen soll:

```
3: var target:MovieClip = createEmptyMovieClip('target',1);
```

Nun ist es soweit. Alle Vorbereitungen sind getroffen. Erzeugen Sie jetzt eine Instanz der `SlideShow`-Klasse, der Sie `target` und `items` übergeben:

```
4: var slideShow:SlideShow = new SlideShow(target,items,4);
```

Sie übergeben auch bereits den Parameter, der in Sekunden angibt, wie lange ein Bild angezeigt werden soll, bevor es mit dem nächsten überblendet wird. Dieser Parameter (hier 4) hat jedoch zum derzeitigen Stand der Programmierung keinen Effekt, da Sie die Überblendung erst in den nächsten Schritten erledigen werden.

Testen Sie nun Ihren Flash-Film mit `[Strg]/[⌘]+[↵]`. Wie Sie sehen, erscheinen die Bilder nach einer kurzen Ladezeit hintereinander auf der Bühne. Das Laden funktioniert also.

Schritt 10

Ladevorgang testen

Schritt 11

Film testen

Abbildung 2 ►

Die Bilder werden auf die Bühne geladen.



Schritt 12

Bilder unsichtbar machen

Wechseln Sie nun wieder zu der SlideShow-Klasse. Wie Sie soeben beim Laden der Bilder bemerkt haben, erscheinen die Bilder kurz nacheinander auf der Bühne. Da dies aber nicht Teil der Diashow sein soll, werden Sie nun die Movieclips, in die die Bilder geladen werden, so lange unsichtbar machen, bis alle Bilder geladen sind. Erst dann sollen die Bilder sichtbar sein, und die Diashow kann beginnen. Nutzen Sie zu diesem Zweck die Vorteile der von Ihnen programmierten ClipLoader-Klasse, indem Sie der `loadClip`-Methode ein Initialisierungsobjekt übergeben und die `_visible`-Eigenschaft des Movieclips auf `false` setzen. Dies erledigen Sie in der letzten Zeile der `loadItems`-Methode in der SlideShow-Klasse:

```
clipLoader.loadClip(itemurls[i],item,{_visible:false});
```

Schritt 13

Film erneut testen

Testen Sie nun wieder Ihren Flash-Film mit `Strg`/`⌘` + `↵`. Sie werden keine Bilder mehr sehen (siehe Abbildung 3) – doch genau das war das Ziel des letzten Schritts.

Schritt 14

Geladene Bilder zählen

In Schritt 8, Zeile 7 haben Sie die SlideShow-Klasse bzw. die Instanz dieser Klasse dem ClipLoader als Listener hinzugefügt. Der ClipLoader erbt von `MovieClipLoader` und sendet somit das Event `onLoadInit()`, sobald auf die Eigenschaften der geladenen Datei zugegriffen werden kann.

Die Diashow soll erst dann mit der Anzeige der Bilder beginnen, wenn diese vollständig geladen sind. Andernfalls könnte es dazu führen, dass die Überblendung einsetzt, ohne dass das nächste Bild geladen ist.

Schreiben Sie also unterhalb der `loadItems`-Methode die `onLoadInit`-Methode:

```
private function onLoadInit():Void
{
    numItemsLoaded++;
    if(numItemsLoaded == items.length){
        startSlideshow();
    }
}
```

Bei jedem Aufruf von `onLoadInit()` zählen Sie `numItemsLoaded` herauf und prüfen in einer `if`-Abfrage, ob dieser Wert der Länge des `items`-Arrays entspricht und somit alle Dateien geladen wurden. Sollte dies der Fall sein, starten Sie die Slideshow mit dem Aufruf einer Methode `startSlideshow()`, die Sie im folgenden Schritt programmieren werden.

Die Slideshow wird gestartet, wenn alle Dateien geladen sind. Schreiben Sie dazu die Methode `startSlideshow()` gleich unterhalb der `onLoadInit`-Methode:

```
private function startSlideshow() : Void
{
    currentItem = 0;
    TransitionManager.start(items[currentItem], {type:Fade,
    direction:Transition.IN, duration:seconds/2, easing:
    Strong.easeOut});
}
```

Die Slideshow soll zu Beginn das erste Element des `items`-Array anzeigen. Zu diesem Zweck setzen Sie in der dritten Zeile die Klassenvariable `currentItem` auf 0. `currentItem` wird die Position im `items`-Array und somit die Bilder referenzieren. In der nächsten Zeile nutzen Sie den `TransitionManager`, um das erste Bild einzublenden. Damit die Dauer der Animation nicht größer ist als die des Intervalls, in dem die Bilder wechseln, setzen Sie das `duration`-Attribut auf die Hälfte dieser Zeit: `seconds/2`.

Testen Sie jetzt Ihren Flash-Film. Nach einer kurzen Zeit, in der die Bilder im Hintergrund geladen werden, blendet sich das erste Bild ein.

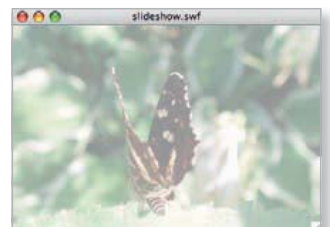


▲ **Abbildung 3**

Die Bilder werden nicht mehr auf der Bühne angezeigt – geladen werden sie dennoch.

Schritt 15

Slideshow starten



▲ **Abbildung 4**

Das erste Bild der Diashow wird eingeblendet.

Schritt 16

Film testen

Schritt 17

Intervall erzeugen

Nun müssen Sie das Intervall erzeugen, über das eine Methode zum Wechseln der Bilder aufgerufen werden soll. Erweitern Sie zu diesem Zweck die `startSlideShow`-Methode um folgende Codezeilen:

```
clearInterval(interval);
interval = setInterval(this,"showNext",seconds*1000);
```

In der ersten Zeile löschen Sie zuerst das Intervall `interval`. Dies geschieht zur Sicherheit, falls `interval` aus unerwarteten Gründen bereits aktiv sein sollte.

In der zweiten Zeile starten Sie nun das Intervall und speichern es in der Klassenvariablen `interval`. Das Intervall ruft eine Methode `showNext()` auf, die Sie im nächsten Schritt programmieren werden. Der dritte Parameter der `setInterval`-Methode bestimmt die Dauer des Intervalls. Da diese Zeit in Millisekunden angegeben werden muss, müssen Sie `seconds` übergeben und mit 1000 multiplizieren.

Schritt 18

Bildwechsel programmieren

In diesem Schritt programmieren Sie die Methode `showNext()`, die durch das im vorherigen Schritt erzeugte Intervall in einer bestimmten Zeit immer wieder aufgerufen wird. Ziel ist es, das aktuelle Bild auszublenden und das nächste Bild einzublenden. Geben Sie dazu folgende Zeilen direkt unter der `startSlideShow`-Methode ein:

```
1: private function showNext():Void
2: {
3:     var lastItem:Number = currentItem;
4:     currentItem = currentItem < items.length-1 ?
        ++currentItem : 0;
5:     TransitionManager.start(items[currentItem], {type:Fade,
        direction:Transition.IN, duration:seconds/2,
        easing:Strong.easeOut});
6:     TransitionManager.start(items[lastItem], {type:Fade,
        direction:Transition.OUT, duration:seconds/2,
        easing:Strong.easeOut});
7: }
```

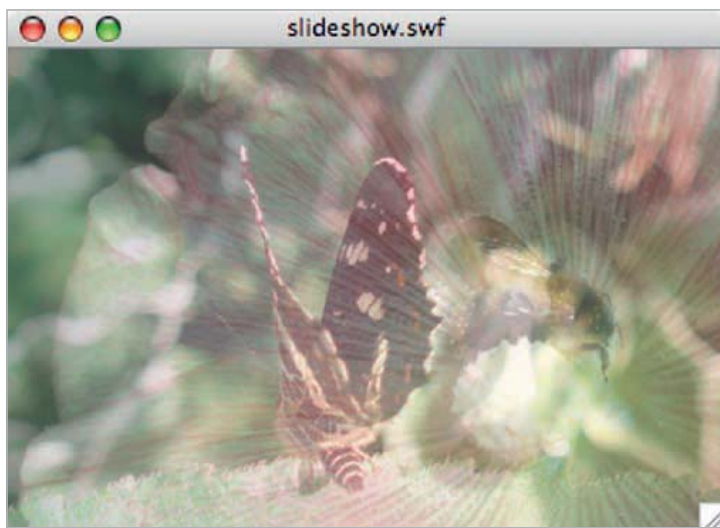
In Zeile 3 erzeugen Sie eine numerische Variable `lastItem`. Diese erhält den Wert von `currentItem` und wird in dem `items`-Array bei der Übergabe an den `TransitionManager` auf das aktuell angezeigte Bild verweisen. Darauf folgend wird `currentItem` um 1 erhöht.

Sollte `currentItem` dann allerdings größer als die Länge des `items`-Arrays minus 1 sein und somit auf ein undefiniertes Feld in dem Array verweisen, wird `currentItem` auf 0 gesetzt. `currentItem` wird in `items` das nächste anzuzeigende Bild bei der Übergabe an den `TransitionManager` referenzieren.

In Zeile 5 und 6 übergeben Sie schließlich dem `TransitionManager` den nächsten `Movieclip` zum Einblenden und den letzten `Movieclip` zum Ausblenden. Beide `Movieclips` werden gleichzeitig ein- und ausgeblendet.

Herzlichen Glückwunsch, die `SlideShow`-Klasse ist damit fertig und einsatzbereit!

Testen Sie nun Ihren Flash-Film. Wie Sie sehen, sind die Bilder nach einer kurzen Weile geladen und die Diashow beginnt. Die Bilder werden nacheinander eingeblendet, und nach dem letzten Bild startet alles wieder von vorne.



Schritt 19

Ergebnis testen

Erweiterung

Überlegen Sie, wie Sie die `SlideShow`-Klasse noch erweitern können. Wie wäre es, damit das nächste Bild von außen zu bestimmen? Eine Vor- und Zurück-Funktionalität einzubauen? Probieren Sie auch die weiteren Überblendungsmöglichkeiten der `TransitionManager`-Klasse aus.

◀ Abbildung 5

Die Diashow ist fertig.

XML

Was ist XML?

Die **Extensible Markup Language**, oder kurz XML, ist eine Auszeichnungssprache, die in strukturierter, textbasierter Form komplexe Daten und deren Zusammenhänge darstellt. Da XML inzwischen fast überall zum Standard geworden ist, bietet es sich im Besonderen für das Speichern und den Austausch von Daten zwischen zwei oder mehreren unterschiedlichen Anwendungen an. Hier ein einfaches Beispiel einer XML-Struktur:

```
<buch>
  <autor>Michael Ende</autor>
  <titel>Momo</titel>
  <verlag>Thienemann</verlag>
</buch>
```

Wie Sie sehen, können Sie auf den ersten Blick die Inhalte sowie die Struktur der vorgegebenen Daten erkennen. Das ist ein weiterer Vorteil von XML: Es ist ohne Probleme vom Menschen zu deuten.

Bei dem Aufbau von XML-Daten spricht man oft von einer Baumstruktur, wobei jedes Element, die sogenannten Knoten, auch als Ast bzw. Blatt des Baumes angesehen werden können. In dem obigen Beispiel ist `<buch>` der grundlegende Knoten, oder besser: **root**-Knoten (root: dt. Wurzel). Damit ein XML-Dokument als wohlgeformt gilt, benötigt es unter anderem genau einen root-Knoten, der alle übrigen Knoten beinhaltet. Weiterhin muss jeder geöffnete Knoten auch wieder geschlossen werden.

Zwei weitere wichtige Begriffe sind die des **child** (dt.: Kind) und des **parent** (dt.: Elternteil), die die Abhängigkeiten der einzelnen Knoten voneinander beschreiben. Das `<autor>`-Element würde im vorherigen Beispiel als child des `<buch>`-Knotens bezeichnet. `<buch>` wäre dementsprechend parent des `<autor>`-Knotens. Auch Flash arbeitet intern mit diesen Bezeichnungen.

XML in Flash

Flash bietet Ihnen seit der Version 5 die Möglichkeit, mit XML zu arbeiten. ActionScript 1.0 und 2.0 stellen zu diesem Zweck die eingebaute XML-Klasse zur Verfügung, die es Ihnen ermöglicht, über Variablen wie z. B. `firstChild` oder `nextSibling` auf die einzelnen XML-Knoten (engl.: nodes) und deren Eigenschaften (engl.: attributes) zuzugreifen.

Wenn Sie also auf den Autor, den Titel und den Verlag des Buches im obigen Beispiel zugreifen wollen, können Sie die XML-Klasse zu Hilfe nehmen. In ActionScript 1.0 oder 2.0 würde das wie folgt aussehen:

```
var myXml:XML = new XML("<buch><autor>Michael Ende</autor>  
<titel>Momo</titel><verlag>Thienemann</verlag></buch>");  
trace(myXml.firstChild.childNodes[0].firstChild);  
trace(myXml.firstChild.childNodes[1].firstChild);  
trace(myXml.firstChild.childNodes[2].firstChild);
```

Sie würden bei Ausführung dieses Skripts innerhalb der Flash-Autorenumgebung »Michael Ende«, »Momo« und »Thienemann« im Ausgabefenster angezeigt bekommen. Die Verarbeitung der XML-Daten über die DOM-basierte XML-Klasse ist demnach möglich, aber der Zugriff auf die Elemente recht kompliziert. Abhilfe schafft hier die rundum erneuerte XML-Verarbeitung in ActionScript 3.0.

XML mit ActionScript 3.0

Die Verarbeitung von XML-Daten innerhalb einer Flash-Anwendung übernimmt seit AS3 die generalüberholte XML-Klasse. Diese setzt das **ECMA-Script for XML** (kurz: E4X) um und steigert damit die Benutzerfreundlichkeit und Flexibilität der Verarbeitung.

Das obige AS2-Beispiel würde ab sofort in AS3 so aussehen:

```
var myXml:XML = <buch><autor>Michael Ende</autor><titel>Momo  
</titel><verlag>Thienemann</verlag></buch>;  
trace(myXml.autor[0]);  
trace(myXml.titel[0]);  
trace(myXml.verlag[0]);
```

Auch XML-Anfänger sehen auf den ersten Blick, dass der Zugriff auf die einzelnen Elemente der XML-Struktur sich wesentlich vereinfacht hat. Sie können nun ohne Umwege direkt über den Namen des Knotens an dessen Inhalte kommen – kein kompliziertes `firstChild.childNodes[0].firstChild` mehr. Zudem wird XML in ActionScript nicht mehr einfach als Zeichenkette behandelt, sondern ist ab sofort ein nativer Datentyp und kann ganz ohne Anführungszeichen geschrieben werden.

Abschließend gilt zu sagen, dass das Arbeiten mit XML-formatierten Daten sich mit AS3 im Vergleich zu AS2 um ein Vielfaches verbessert hat.

DOM

DOM oder auch **Document Object Model** ist keine weitere Programmiersprache, sondern eine Schnittstelle, die es dem Entwickler ermöglicht, auf die Elemente z.B. innerhalb eines HTML- oder XML-Dokuments zuzugreifen.

Ausprobieren!

Versuchen Sie einfach mal, den folgenden Workshop nach ActionScript 3.0 zu portieren. Sie werden die Unterschiede schnell erkennen und schätzen lernen.

XML-Menü

Ein Menü extern über XML steuern

```
class XmlMenu{
    var xml:XML;
    var menuTexts:Array;
    var menuActions:Array;
    var contentMc:MovieClip;
    var menuItemMc:String;
    var menuHolder:MovieClip;
    public function XmlMenu(base:Object, contentMc:MovieClip,
        menuItemMc:String, x:Number, y:Number){
        this.xml = new XML();
        this.xml.ignoreWhite = true;
        this.menuTexts = new Array();
        this.menuActions = new Array();
        this.contentMc = contentMc;
        this.menuItemMc = menuItemMc;
        this.menuHolder = base.createEmptyMovieClip("menuHolder",
            base.getNextHighestDepth());
        this.menuHolder._x = x;
        this.menuHolder._y = y;
        getXMLData();
    }
    private function getXMLData():Void{
        var scope:Object = this;
        this.xml.onLoad=function(success){
            if(success){
                for (var i:Number=0; i<scope.xml.firstChild.
```

So schön Flash auch ist – es hat auch Nachteile. Einer davon ist die oftmals starre Beschaffenheit einer Flash-Anwendung. Vergleichen Sie es beispielsweise mit HTML, wo Sie mit relativ geringem Aufwand neue Inhalte nachträglich einfügen können. Demgegenüber ist das mit einer swf-Datei deutlich aufwendiger und oft ohne Flash gar nicht möglich. Doch es gibt Abhilfe.

In diesem Workshop lernen Sie, das komplette Menü einer Flash-Anwendung in eine XML-Datei auszulagern, so dass Flash nicht mehr benötigt wird, um Menüpunkte einzufügen, zu bearbeiten oder zu löschen.

Zielsetzungen:

- Inhalte bereitstellen
- XML-Daten auslesen
- Menüelemente mit Funktionalität ausstatten

[Ergebnis: 28_XML-MENUE • XML_MENUE.SWF]

Schnee

Sonnenuntergang

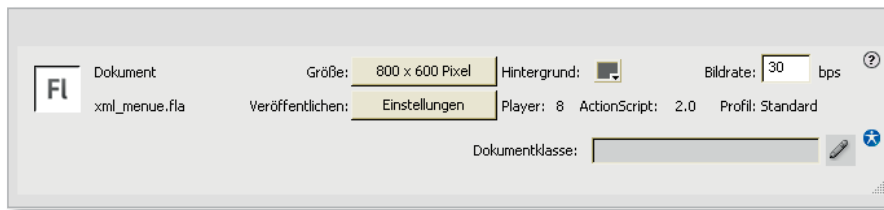
Feder

Typo

Herzlich Willkommen

Im ersten Schritt legen Sie einen neuen Ordner auf Ihrer Festplatte an und nennen ihn XML-MENÜE. Hier werden Sie die von Ihnen erstellten Beispiele zu diesem Workshop abspeichern.

Erstellen Sie eine neue Flash-Datei (ActionScript 2). Wählen Sie eine Bühnengröße von 800x600 Pixel, eine BILDRATE von 30bps und eine dunkelgraue Hintergrundfarbe. Die Flash-Datei fungiert hier als beispielhafte Flash-Anwendung, weshalb die Einstellungen auch nur exemplarisch sind.



Das XML-Menü wird den ClipLoader aus dem gleichnamigen Workshop benutzen (21_CLIPLOADER), weshalb die Datei CLIPLOADER.AS im Verzeichnis vorhanden sein muss. Außerdem werden einige Dateien benötigt, die vom Menü dynamisch als Inhalte geladen werden. Dies können Bilder oder swf-Dateien sein. Kopieren Sie die Bilddateien SONNENUNTERGANG.JPG und FEDER.PNG von der Buch-DVD in Ihr Verzeichnis. Als swf-Dateien verwenden wir einfach swf-Dateien aus anderen Workshops: HANDSCHREIB-EFFEKT.SWF und SCHNEESIMULATION.SWF. Sie liegen bereits im Ordner dieses Workshops auf der Buch-DVD.

Erstellen Sie in demselben Verzeichnis eine ActionScript-Datei mit dem Namen XMLMENU.AS. Deklarieren Sie folgende Variablen:

```
1: class XmlMenu{
2:     var xml:XML;
3:     var menuTexts:Array;
4:     var menuActions:Array;
5:     var contentMc:MovieClip;
6:     var menuItemMc:String;
7:     var menuHolder:MovieClip;
8: }
```

Schritt 1

Vorbereitungen

Schritt 2

Flash-Datei anlegen

◀ **Abbildung 1**
Die Dokumenteigenschaften können Sie wie in der Abbildung gezeigt festlegen – oder eigene verwenden.

Schritt 3

Benötigte Dateien bereitstellen

Schritt 4

ActionScript-Datei anlegen

Die Variable `xml` ist eigentlich selbsterklärend – sie wird für eine Instanz der Klasse `XML` benötigt, die das Laden von XML-Dateien ermöglicht. Die beiden Arrays (Zeilen 3 und 4) dienen als Datencontainer für die aus der XML-Datei ausgelesenen Daten. Die Variable `contentMc` steht für den Movieclip, in den der Inhalt geladen wird. Schließlich wird noch der Name eines Movieclips (Zeile 6) benötigt, der dann ein einzelnes Menüelement grafisch repräsentiert, sowie ein Movieclip (Zeile 7), der die einzelnen Menüelemente enthält.

Schritt 5 Der Konstruktor

Der Konstruktor initialisiert die Variablen und startet das Auslesen der XML-Datei. Er wird, wie immer, direkt unter den Variablendeklarationen eingefügt:

```
1: public function XmlMenu(base:Object, contentMc:MovieClip,  
    menuItemMc:String, x:Number, y:Number){  
2:     this.xml = new XML();  
3:     this.xml.ignoreWhite = true;  
4:     this.menuTexts = new Array();  
5:     this.menuActions = new Array();  
6:     this.contentMc = contentMc;  
7:     this.menuItemMc = menuItemMc;  
8:     this.menuHolder = base.createEmptyMovieClip("menuHolder",  
        base.getNextHighestDepth());  
9:     this.menuHolder._x = x;  
10:    this.menuHolder._y = y;  
11:    getXMLData();  
12: }
```

XML

XML ist eine Auszeichnungssprache, die Daten und deren Zusammenhänge in textbasierter Form darstellt. XML ist besonders gut zum Austausch von Daten zwischen verschiedenen Anwendungen geeignet. Ein einzelnes Element wird hierbei als Knoten bezeichnet.

Als Parameter erwartet der Konstruktor das dem Menü übergeordnete Objekt, das oftmals (wie auch hier) die Bühne selbst sein wird. Es wird in Zeile 8 verwendet, um den Container für die einzelnen Menüelemente zu erstellen. Der zweite Parameter `contentMc` ist der Container für den Inhalt (vergleiche Schritt 4). `menuItemMc` ist der Name (genauer gesagt der Bezeichner) für einen beliebigen Movieclip, der als Hintergrund für die einzelnen Menüelemente fungiert. Er wird in der fla-Datei erstellt. Schließlich werden noch die Koordinaten des Menüs übergeben.

Die Zeilen 2 bis 7 initialisieren die Variablen, wobei entweder neue Objekte erstellt oder die Konstruktorparameter zugewiesen werden.

Die einzige Ausnahme bildet hier Zeile 3, in der festgelegt wird, dass Textknoten ignoriert werden, wenn diese nur aus Leerzeichen bestehen. In

den Zeilen 9 und 10 wird das Menü positioniert. Zeile 11 schließlich ruft die Funktion zum Laden der XML-Daten auf und startet das eigentliche Prozedere.

Die folgende Funktion liest die XML-Daten aus und ruft während des Auslesens eine Funktion zum Erstellen einzelner Menüpunkte auf. Ergänzen Sie sie direkt unter dem Konstruktor:

```
1: private function getXMLData():Void{
2:     var scope:Object = this;
3:     this.xml.onLoad=function(success){
4:         if(success){
5:             for (var i:Number=0; i<scope.xml.firstChild.
               childNodes.length; i++) {
6:                 scope.menuTexts[i]= scope.xml.firstChild.
               childNodes[i].attributes.name;
7:                 scope.menuActions[i]= scope.xml.firstChild.
               childNodes[i].attributes.action;
8:             }
9:             scope.drawMenu();
10:        }
11:    }
12:    this.xml.load("menu.xml");
13: }
```

Zunächst wird eine Referenz auf die Klasse in einer Variablen `scope` zwischengespeichert, um innerhalb der `onLoad`-Methode darauf zugreifen zu können. In dieser wird zunächst überprüft, ob das Laden erfolgreich war (`if(success)`).

Dann wird in den Zeilen 5 bis 8 die XML Datei ausgelesen und die so ermittelten Daten in die entsprechenden Arrays geschrieben. Hierbei werden die Beschriftungen der Menüeinträge im Array `menuTexts` und der Name der zu ladenden Datei im Array `menuActions` gespeichert.

In Zeile 9 schließlich wird die Funktion `drawMenu()` aufgerufen, die das eigentliche Zeichnen des Menüs übernimmt. In Zeile 12 wird die XML-Datei geladen, damit die `onLoad`-Methode überhaupt ausgeführt wird.

Um das Menü zu zeichnen, definieren Sie eine eigene Methode. Geben Sie folgenden Code unter der in Schritt 6 eingefügten Funktion ein:

Schritt 6

XML-Daten auslesen

XML und Flash

Wenn Sie mehr über Flash und XML erfahren wollen, lesen Sie den Exkurs »XML« in diesem Buch.

Schritt 7

Das Menü zeichnen

```

1: private function drawMenu():Void{
2:     var tf:TextFormat = new TextFormat();
3:     tf.font = "Verdana";
4:     tf.size = 12;
5:     tf.color = 0xffffffff;
6:     for(var i:Number = 0; i<this.menuTexts.length; i++){
7:         var menuItem:MovieClip = menuHolder.attachMovie(this,
            menuItemMc, "menuItemMc"+i, menuHolder.
            getNextHighestDepth());
8:         menuItem._y = i*(menuItem._height+5);
9:         var labelText:TextField = menuItem.
            createTextField("text", menuItem.getNextHighestDepth(),
            10, 5, menuItem._width-10, menuItem._height-5);
10:        labelText.text = menuTexts[i];
11:        labelText.setTextFormat(tf);
12:        this.initActions(menuItem, i);
13:    }
14: }

```

In den Zeilen 2 bis 5 wird ein Textformat für die Menüpunkte definiert. In Zeile 6 wird eine `for`-Schleife gestartet, die den folgenden Anweisungsblock so oft durchläuft, wie das Array `menuTexts` Elemente hat.

Dann werden die Menüelement-Movieclips erzeugt (Zeile 7). In Zeile 8 werden sie positioniert, wobei ein Abstand von fünf Pixel festgelegt wird.

In den Zeilen 9 bis 11 wird ein Textfeld erzeugt, positioniert und mit Text und Textformat versehen. In Zeile 12 wird eine Funktion `initActions()` aufgerufen. Diese weist den Menüelementen den zu ladenden Inhalt zu.

Schritt 8

Die Menüelemente mit Funktionalität ausstatten

Für das Ausstatten der Menüelemente mit Funktionalität stehen zwei Funktionen bereit. Die Funktion `initActions()` weist jedem Menüelement das entsprechende Verhalten zu. Fügen Sie sie direkt unter der eben erstellten Funktion `drawMenu()` ein:

```

1: private function initActions(menuItem:MovieClip,
    i:Number):Void{
2:     var scope:Object = this;
3:     menuItem.action = this.menuActions[i];
4:     menuItem.onRelease = function(){
5:         scope.loadNewContent(this.action);
6:     }
7: }

```

Da die Funktion aus einer Schleife heraus aufgerufen wird, steht der Parameter `i` immer für den jeweils richtigen Index im Array `menuActions`. Dem Menüelement wird als Eigenschaft der Dateiname der zu ladenden Datei übergeben (Zeile 3). Dann wird die Funktion `onRelease` definiert. In dieser wird die Funktion `loadNewContent()` für das eigentliche Laden von Inhalten mit dem Dateinamen als Parameter aufgerufen (Zeile 5). Diese Funktion sieht dann folgendermaßen aus und wird unter der eben erstellten eingefügt:

```
private function loadNewContent(newContent:String):Void{
    var cl:ClipLoader = new ClipLoader();
    cl.loadClip(newContent, this.contentMc);
}
```

In dieser Funktion wird mit Hilfe des ClipLoaders der neue Inhalt geladen.

Das war eine ganze Menge Quelltext ohne erfrischenden Zwischentest. Aber nun ist es soweit. Legen Sie eine XML-Datei mit dem Namen `MENU.xml` in unserem Workshop-Verzeichnis `XML-MENUE` an. Geben Sie folgenden Code ein:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<menu>
<menuitem name="Schnee" action="Schneesimulation.swf" />
<menuitem name="Sonnenuntergang" action="Sonnenuntergang.jpg" />
<menuitem name="Feder" action="Feder.png" />
<menuitem name="Typo" action="Handschreibeffekt.swf" />
</menu>
```

Wechseln Sie in die Flash-Datei, und versehen Sie Bild 1 der Hauptzeitleiste mit folgendem Code:

```
System.useCodepage=true;
var xml:XmlMenu = new XmlMenu(this, content_mc, "menu", 10, 10);
```

Die erste Zeile vermeidet, grob gesagt, Probleme mit Umlauten. In der zweiten Zeile steht der Parameter `content_mc` für einen Movieclip, in den der Inhalt geladen wird. Diesen müssen Sie natürlich auch anlegen und mit dem Instanznamen »`content_mc`« versehen. Der Parameter `"menu"`

Schritt 9

Die XML-Datei

Ein Hinweis zum Layouten

Prinzipiell könnte sich ein solcher Movieclip natürlich von der Größe her dynamisch dem Text anpassen. So müssten Sie nicht selbst dafür Sorge tragen, dass der Text auch genug Platz hat. Aber bedenken Sie, dass es damit mehr oder weniger dem Zufall überlassen bliebe, wie groß die einzelnen Menüeinträge sind – das Layout wäre schnell »zerschossen«. Eine feste Größe löst dieses Problem, führt aber natürlich dazu, dass Sie keine beliebig langen Worte verwenden können. Layouten erfordert insbesondere in Verbindung mit dynamischen Inhalten stets sorgfältige Vorüberlegungen.

steht für einen Movieclip, der die grafische Darstellung eines einzelnen Menüelements beinhaltet.

Legen Sie also ganz nach Geschmack einen geeigneten Movieclip in der Bibliothek an. Bedenken Sie hierbei, dass der Text eines Menüeintrags (etwa »Home«, »Referenzen« etc. für ein Portfolio) vollständig in diesen Movieclip passen sollte – ansonsten lassen Sie Ihrer Kreativität freien Lauf.

Sie müssen diesen Movieclip nicht auf der Bühne platzieren, ihm jedoch den Bezeichner »menu« geben. Führen Sie dazu einen Rechtsklick bzw. **Ctrl**-Klick auf das Symbol in der Bibliothek aus, und wählen Sie im Kontextmenü **VERKNÜPFUNG**. Im folgenden Dialogfenster können Sie den Bezeichner angeben.

Nun können Sie endlich über **Strg**/**⌘** + **↵** kompilieren. Klicken Sie sich durch Ihre neue Mini-Applikation – Sie sehen in jedem Menüpunkt den in der XML-Datei festgelegten Inhalt.

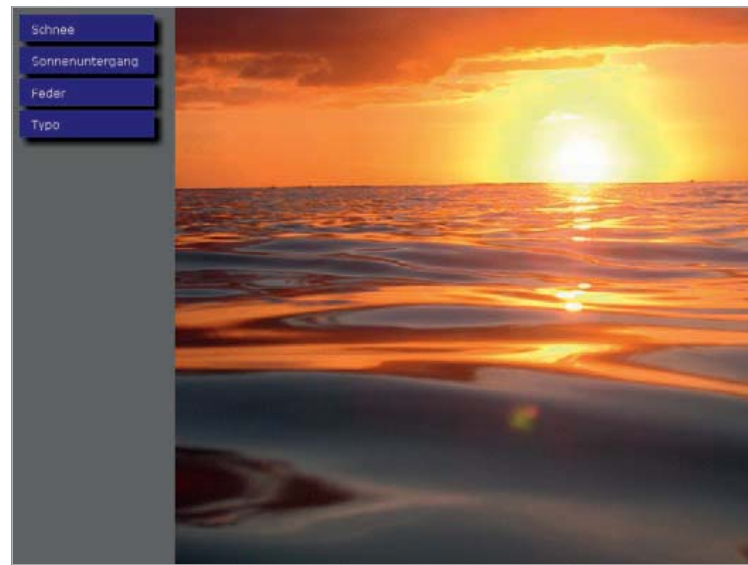


Abbildung 2 ►
Das fertige Menü

Flash Remoting

Flash Remoting können Sie als Bindeglied zwischen Client (Flash-Anwendung) und Server verstehen. Es ermöglicht den direkten Datenaustausch zwischen Flash- und Server-Applikationen. Ursprünglich stand Flash Remoting nur Benutzern der teuren Serverlösungen von Macromedia zur Verfügung (ColdFusion, JRun). Inzwischen wurden auch diverse Open-Source-Lösungen für PHP (AMFPHP) oder Java (OpenAMF) und für viele andere Programmiersprachen entwickelt.

Flash Remoting ist eine serverseitige Technologie, die sich in vorhandene Applikationsserver integriert und ein Gateway zwischen dem Flash Player und den Serverdiensten über **Remote Services** herstellt. Ein Service kann aus einer einfachen ColdFusion-Seite, einem PHP- bzw. Perl-Skript oder auch einer Java-Klasse bestehen. Das Remoting-Gateway ermöglicht es Ihnen, die Services direkt aus Ihrer Flash-Anwendung heraus zu adressieren und liefert Ihre Daten auf demselben Weg auch wieder zurück.

Die grundsätzlichen Funktionen des Remoting-Gateways können wie folgt zusammengefasst werden:

1. Es verarbeitet alle Anfragen des Flash Players an die Remote Services, wobei sich die Services auf demselben Server befinden können oder aber als webbasierte Dienste auf weiteren Applikationsservern.
2. Es serialisiert und deserialisiert die übertragenen Daten zwischen Flash Player und den Services. Unter Serialisierung versteht man die Datenkonvertierung in einen einfachen binären Datenstrom. Diese Konvertierung spart Bandbreite, da sie ein wesentlich kleineres Datenaufkommen als andere String-basierte Übertragungen verursacht. Um eine möglichst schnelle und unkomplizierte Kommunikation zwischen der Flash-Anwendung und den Remote Services zu gewährleisten, serialisiert Flash Remoting die Meldungen bidirektional als **Action Message Format** (AMF), das ein auf dem **Simple Object Access Protocol** (SOAP) basierendes Binärformat ist.
3. Es führt eine Typkonvertierung zwischen Flash-Datentypen und den Services durch. So können beispielsweise Objekte aus dem Flash-Film als PHP-Objekte auf dem Applikationsserver verwendet werden und umgekehrt.

In dem passenden Workshop »Flash-Gästebuch« zu diesem Exkurs benutzen Sie ein **Value Object** (VO), das genau auf dieser Funktionalität des Remoting-Gateways aufbaut.

Wenn Sie Flash Remoting mit Flash 8 verwenden wollten, mussten Sie, bevor Sie beginnen konnten, noch weitere Remoting-Komponenten installieren. Mit Flash CS3 wurde auch diese kleine Hürde abgeschafft und ActionScript 3 unterstützt Remoting nun von Anfang an.

Die Benutzung von Flash Remoting ist momentan leider noch nicht auf mobilen Endgeräten möglich. Das gerade in der Version 1 erschienene **SWX** (<http://swxformat.org>) wäre hier eine gute Alternative.

Flash-Gästebuch

Flash Remoting für dynamische Inhalte

Wie oft wollten Sie schon dynamische Inhalte in Ihre eigene Webseite integrieren, sind aber bisher immer daran gescheitert, die erste Hürde zu nehmen? Dieser Workshop erklärt Ihnen anhand eines selbst erstellten Gästebuchs, wie Sie mit Hilfe von Flash Remoting dynamische Inhalte aus einer Datenbank auslesen und in Ihrem Flash-Film anzeigen. Nun steht Ihnen die bunte Welt des Web 2.0 offen ...

Zielsetzungen:

- Die notwendigen Grafiken erstellen
- Komponenten einsetzen
- Flash-Formular anlegen
- Arbeiten mit Flash Remoting
- Erstellen des Gästebuchs

[Ergebnis: 29_FLASH-GAESTEBUCH • GAESTEBUCH.HTML]

```
package
{
    import flash.display.MovieClip;
    import flash.text.TextField;
    import flash.net.NetConnection;
    import flash.net.Responder;
    import flash.net.registerClassAlias;
    import flash.events.MouseEvent;
    import flash.events.Event;
    public class Main extends MovieClip
    {
        private const GATEWAY_URL:String =
            "http://flashfastforward.de/amfphp/gateway.php";
        private var conn:NetConnection;
        private var posts_mc:MovieClip;
        public function Main()
        {
```

Gästebuch

Name:

Webseite (optional) (ohne http://):

Nachricht:

absenden

Bitte geben Sie einen Namen ein.

Schritt 1 Vorbereitungen

Im ersten Schritt legen Sie einen neuen Ordner auf Ihrer Festplatte an und nennen ihn GAESTEBUCH. Hier werden Sie die von Ihnen erstellten Beispiele zu diesem Workshop abspeichern.

Schritt 2 Erstellen der Flash-Datei

Wechseln Sie in die Flash-Autorenumgebung, und erstellen Sie über **[Strg]/[N]** eine neue FLASH-DATEI (ACTIONSCRIPT 3.0) ❶ Bestätigen Sie mit **Ok** ❷.

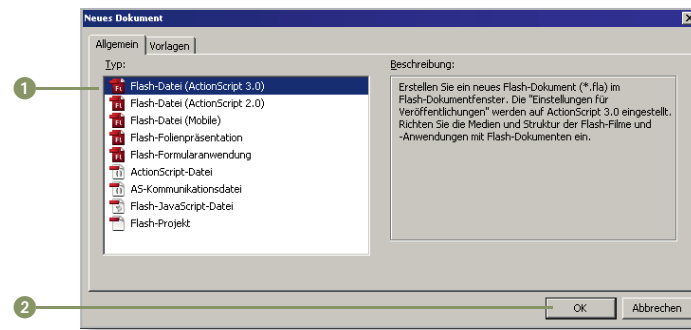


Abbildung 1 ►
Eine neue Flash-Datei vom Typ ActionScript 3 anlegen

Diese Datei speichern Sie unter dem Namen GAESTEBUCH.FLA in dem in Schritt 1 erstellten Ordner ab.

Schritt 3 Anpassen der Dokumenteigenschaften

Da Sie in diesem Workshop etwas mehr Platz auf der Bühne benötigen, öffnen Sie mit **[Strg]/[N] + [J]** die DOKUMENTEIGENSCHAFTEN und ändern die GRÖSSE auf 990x600 Pixel ❸ bei einer BILDRATE von 25 Bildern pro Sekunde ❺. Die HINTERGRUNDFARBE belassen Sie bei Weiß (#FFFFFF) ❹. Bestätigen Sie die Anpassungen mit **Ok**.

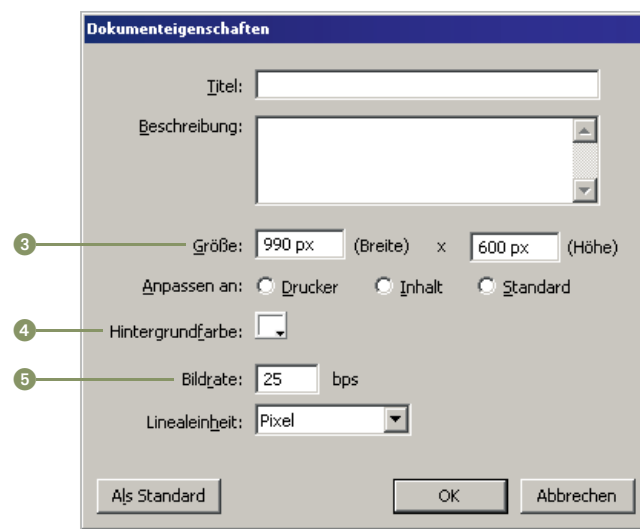
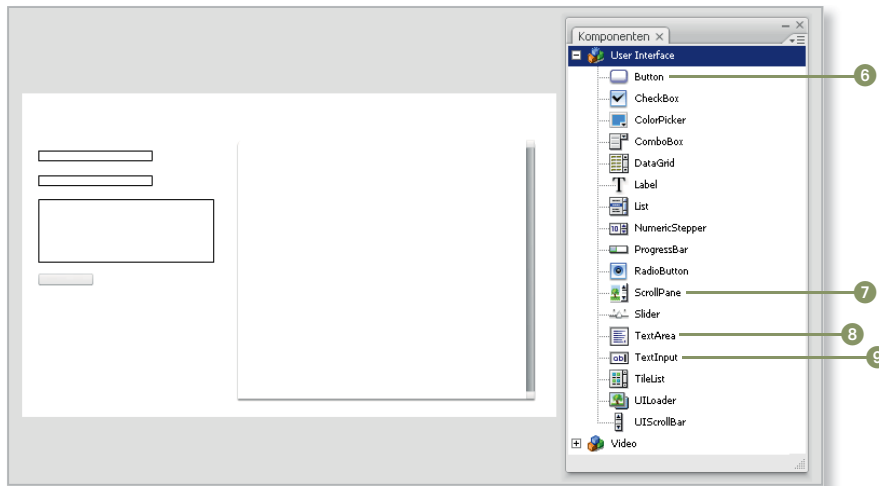


Abbildung 2 ►
Dokumenteigenschaften anpassen

Für das zu erstellende Gästebuch benötigen Sie diverse Flash-eigene Komponenten. Öffnen Sie zu diesem Zweck über FENSTER • KOMPONENTEN das entsprechende Fenster, und ziehen Sie aus dem Unterordner USER INTERFACE folgende Elemente auf die Bühne: BUTTON (1 x) ⑥, TEXTAREA (1 x) ⑧, TEXTINPUT (2 x) ⑨ und SCROLLPANE (1 x) ⑦. Positionieren Sie diese wie im Screenshot gezeigt auf der Bühne.

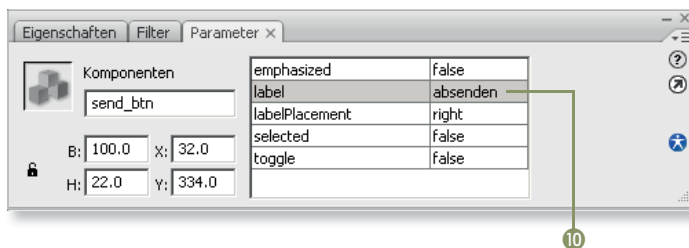


Schritt 4 Komponenten einfügen

◀ **Abbildung 3**
Die Komponenten werden einfach per Drag & Drop auf die Bühne gezogen und arrangiert.

Die beiden TEXTINPUT-Felder sowie die TEXTAREA und der BUTTON sollten auf der linken Seite platziert werden, so dass rechts noch ausreichend Platz bleibt für die 550x480 Pixel große SCROLLPANE.

Geben Sie den beiden TEXTINPUT-Feldern im Eigenschaftensfenster die Instanznamen »name_txt« und »www_txt«. Diese werden später als Eingabefelder für den Namen des Besuchers und seine Webseite dienen. Das mehrzeilige TEXTAREA-Element erhält den Namen »message_txt« und wird später die eigentliche Nachricht des Besuchers beinhalten. Der BUTTON unterhalb der Eingabefelder erhält den Namen »send_btn«. Setzen Sie zusätzlich über das Parameter-Fenster das Label des Buttons auf »absenden« ⑩. Zu guter Letzt benennen Sie die SCROLLPANE mit dem Instanznamen »sp«.



Schritt 5 Instanznamen vergeben

◀ **Abbildung 4**
Über »label« ⑩ bestimmen Sie, was als Beschriftung auf dem Button angezeigt wird.

Schritt 6 Grafiken vervollständigen

Um dem Gästebuch nun noch einen gewissen Rahmen zu geben, fügen Sie eine Seiten-Überschrift hinzu. Diese könnte der Name Ihrer Seite bzw. des Gästebuchs sein. In diesem Beispiel nehmen Sie »Gästebuch«. Schreiben Sie das Wort mit Hilfe des Textwerkzeugs **T** in die linke obere Ecke der Bühne. Es empfiehlt sich, mindestens eine Schriftgröße von ungefähr 48 Pixel zu wählen.

Anschließend fügen Sie die jeweiligen Eingabefeld-Überschriften als statischen Text hinzu. So weiß jeder Besucher, welches Feld welche Eingabe erfordert.

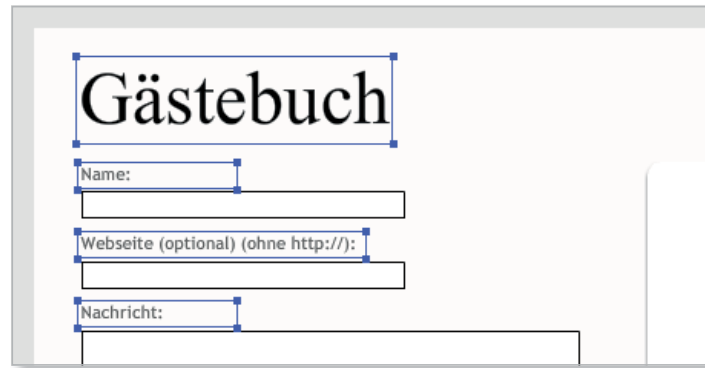


Abbildung 5 ►
Mit Hilfe von Überschriften strukturieren Sie das Gästebuch.

Jetzt fehlt Ihnen lediglich noch ein Textfeld, in dem Sie die möglicherweise auftretenden Fehler ausgeben können. Legen Sie hierfür unterhalb des ABSENDEN-Buttons ein dynamisches Textfeld mit dem Instanznamen »error_txt« an.

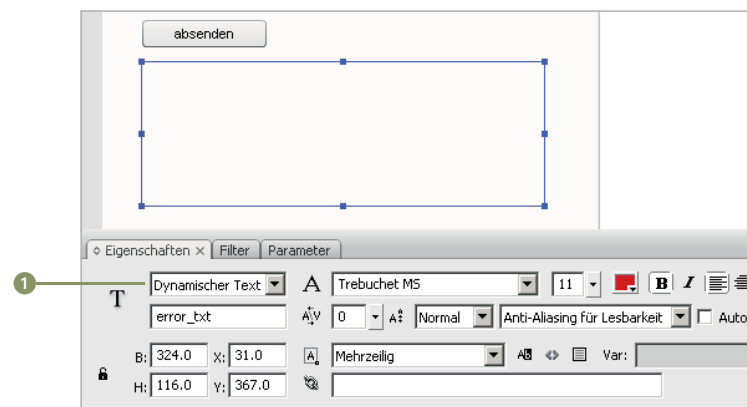


Abbildung 6 ►
Im Eigenschaftensfenster wählen Sie DYNAMISCHER TEXT 1.

Schritt 7 Anlegen der Dokument- klasse

Das gesamte Gästebuch wird mit Hilfe der Dokumentklasse gesteuert. Erzeugen Sie über **Strg**/**⌘**+**N** eine neue ACTIONSCRIPT-DATEI, und fangen Sie mit dem folgenden Klassengerüst an:

```

1: package
2: {
3:     import flash.display.MovieClip;
4:     import flash.text.TextField;
5:     import flash.net.NetConnection;
6:     public class Main extends MovieClip
7:     {
8:         private const GATEWAY_URL:String = "http://
           flashfastforward.de/amfphp/gateway.php";
9:         private var conn:NetConnection;
10:        private var posts_mc:MovieClip;
11:        public function Main()
12:        {
13:        }
14:    }
15: }

```

Die unterschiedlichen Import-Anweisungen in den Zeilen 3 bis 5 benötigen Sie zum einen, weil die Dokumentklasse von `MovieClip` erbt, Sie bereits Textfelder auf der Bühne platziert haben und zum anderen weil die Variable `conn` aus Zeile 9 vom Typ `NetConnection` ist.

In Zeile 8 initialisieren Sie die Konstante `GATEWAY_URL`. Diese speichert die URL des Remoting-Gateways, das Sie im weiteren Verlauf des Workshops ansprechen werden.

Die privaten Instanzvariablen `conn` und `posts_mc` vervollständigen die benötigten Variablen und werden zu einem späteren Zeitpunkt noch zum Einsatz kommen.

In Zeile 11 folgt der bislang noch leere Konstruktor der Klasse. Er wird beim Ausführen der swf-Datei anfangs ein einziges Mal ausgeführt. Im nächsten Schritt werden Sie an dieser Stelle alle notwendigen Vorkehrungen für das Gästebuch treffen.

Damit der Besucher des Gästebuchs Ihr Formular in gewohnter Weise benutzen kann, werden Sie nun im Konstruktor die Tab-Reihenfolge setzen:

```

1: name_txt.tabIndex = 1;
2: www_txt.tabIndex = 2;
3: message_txt.tabIndex = 3;
4: send_btn.tabIndex = 4;

```

URL-Angabe anpassen

In diesem Beispiel benutzen Sie das Gateway des *flash-fastforward.de*-Servers. Hier können Sie natürlich auch eine andere URL eintragen, sollten Sie über eine eigene Installation von AMFPHP verfügen.

Flash Remoting/AMFPHP

Weitere Informationen zu Flash Remoting und AMFPHP finden Sie im Exkurs ab Seite 309 in diesem Buch.

Schritt 8

Formular vorbereiten

Anzeigeliste

Die Anzeigeliste (engl.: DisplayList) stellt die Hierarchie aller grafischen Elemente in Flash dar. Wenn z. B. ein Movieclip der Anzeigeliste hinzugefügt und im sichtbaren Bereich platziert wird, so zeichnet der Flash Player diesen Movieclip.

Mehr zum Thema Anzeigeliste können Sie im Exkurs »Anzeigeliste und attachMovie()« nachlesen.

Diese vier Zeilen bewirken, dass der Benutzer mit der -Taste wie bei einem HTML-Formular von einem Eingabefeld zum nächsten springen kann.

Fügen Sie nun die folgenden Zeilen noch zum Konstruktor hinzu:

```
5: conn = new NetConnection();
6: conn.connect(GATEWAY_URL);
7: posts_mc = new MovieClip();
8: addChild(posts_mc);
```

In Zeile 5 initialisieren Sie die `NetConnection` und stellen in der darauffolgenden Zeile die Verbindung mit dem Remoting-Gateway her. Der `posts_mc`-Movieclip aus Zeile 7 wird später als Container für die anzuzeigenden Gästebucheinträge (engl.: posts) dienen. Per `addChild()` in Zeile 8 fügen Sie diesen Movieclip der Anzeigeliste hinzu.

Speichern Sie über /+ die Klasse als MAIN.AS im Workshop-Ordner ab.

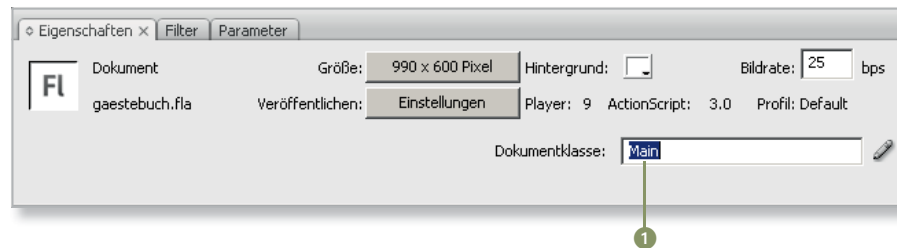
Schritt 9

Dokumentklasse zuweisen

Abbildung 7 ►

Die Dokumentklasse wird im Eigenschaftensfenster des Flash-Films angegeben.

Wechseln Sie zurück zur GAESTEBUCH.FLA, tragen Sie im Eigenschaftensfenster die DOKUMENTKLASSE »Main« ein , und bestätigen Sie die Änderung mit .



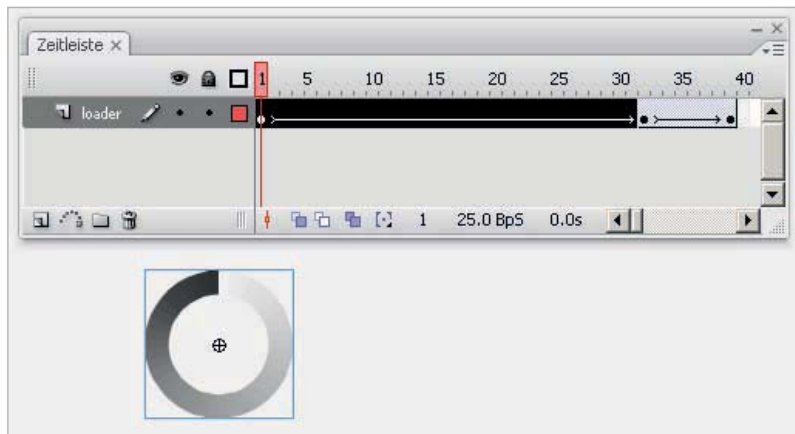
Schritt 10

Modalen Layer erzeugen

Während der Anfragen an das Gateway soll das Gästebuch deaktiviert werden, um Fehler zu vermeiden. Dies erreichen Sie am einfachsten, indem Sie einen Movieclip über den entsprechenden Bereich legen.

Damit der Movieclip nicht die komplette Seite verdeckt, zeichnen Sie ein 960x500 Pixel großes Rechteck  mit der Farbe #666666 und setzen den Alphawert auf 10%. Positionieren Sie das Rechteck so, dass alle Formularelemente verdeckt werden.

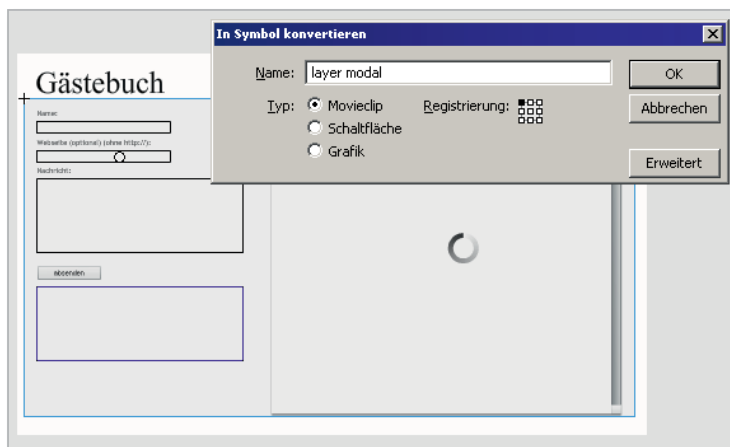
Um dem Besucher zu zeigen, dass während des Einblendens des Layers etwas im Hintergrund passiert, bietet es sich an, eine kleine Animation auf dem Layer anzuzeigen. In der Beispieldatei zu diesem Workshop wird ein sich drehender Kreis als Ladeanimation verwendet.



▲ **Abbildung 8**

Die Animation für den Preloader-Movieclip ist ein einfaches Tweening.

Konvertieren Sie Ihren modalen Layer anschließend mit **[F8]** in ein Symbol vom Typ MOVIECLIP, und geben Sie diesem den NAMEN »layer modal«.



▲ **Abbildung 9**

Der Layer wird in einen Movieclip konvertiert.

Modaler Layer

Der modale Layer soll verhindern, dass der Benutzer die sich unter dem Layer befindlichen Interaktionselemente weiter bedienen kann. Mit ActionScript 2.0 mussten Sie beim Erstellen eines solchen Layers bisher auf eine Schaltfläche zurückgreifen, da diese auf die entsprechenden Ereignisse wie z. B. `MOUSE_OVER` reagiert. In ActionScript 3.0 reicht es aus, einen Movieclip zu verwenden. Obwohl dieser standardmäßig nicht auf die Maus-Ereignisse reagiert, kann der Benutzer im Gegensatz zur vorherigen ActionScript-Version sich nicht mehr durch den Movieclip »durchklicken«.

Preloader-Animation

Sie können entweder die Ladeanimation aus der Beispieldatei übernehmen oder sich eine eigene Ladeanimation auf der Zeitleiste anlegen. In der mitgelieferten Datei gibt es einen Movieclip »pre-loader« mit einem Kreis, der sich über 39 Bilder auf der Zeitleiste einmal um seinen Mittelpunkt dreht. Da Sie nicht auf einmal eine Drehung um 360° vollziehen können, liegt auf Bild 32 ein weiteres Schlüsselbild. Sie können diesen Preloader-Movieclip jederzeit durch eine eigene Zeitleistenanimation ersetzen.

Anschließend geben Sie dem Layer den Instanznamen »modal_mc«.

Im Konstruktor der Dokumentklasse blenden Sie den Layer aus. Geben Sie folgende Anweisung über die in Schritt 8 eingefügten Zeilen ein:

```
modal_mc.visible = false;
```

Schritt 11

Remoting-Gateway installieren

29_FLASH-
GAESTEBUCH •
AMFPHP



Dieser Workshop setzt ein Flash-Remoting-Gateway voraus. Es bleiben Ihnen nun zwei Möglichkeiten, wie Sie weiter vorgehen können: Entweder greifen Sie auf das von uns zur Verfügung gestellte Gateway online zu; oder Sie haben einen eigenen Webserver lokal oder im Internet, auf dem mindestens PHP 4 installiert und eine Datenbank eingerichtet ist.

Wenn Sie sich dazu entschließen, das *flashfastforward.de*-Gateway zu benutzen, und nicht genau erfahren wollen, was auf Seiten des Backends passiert, können Sie direkt zu Schritt 19 springen und dort weitermachen.

Interessiert es Sie aber, wie das Remoting-Gateway zu »installieren« bzw. einzurichten ist, dann lesen Sie hier weiter.

Auf der Buch-DVD finden Sie die Version 1.9 von AMFPHP, das das bekannteste Remoting-Gateway für die Skriptsprache PHP ist. Kopieren Sie den AMFPHP-Ordner aus dem Workshop-Verzeichnis in das root-Verzeichnis Ihres Webserver. Anschließend sollten Sie diese Adresse aufrufen können, um von dort die Bestätigung zu bekommen, dass Ihr Gateway korrekt installiert wurde:

<http://ihre-domain/amfphp/gateway.php>

»ihre-domain« müssen Sie bei einem lokalen Webserver durch »localhost« ersetzen, ansonsten durch die URL Ihrer Website.

29_FLASH-
GAESTEBUCH •
WEBSERVER



Lokalen Webserver installieren

Sollten Sie über keinen eigenen Webserver oder Webspace verfügen, finden Sie auf der DVD zum Buch den XAMPP-Lite. Dies ist ein kompletter Apache Webserver, der ohne großen Aufwand auf Ihrem Rechner »installiert« werden kann. Entpacken Sie einfach die mitgelieferte Datei, und führen Sie anschließend die Datei `SETUP_XAMPP.BAT` per Doppelklick aus. Wenn die Konfiguration ohne Fehler durchläuft, können Sie danach den Apache Webserver über die mitgelieferte `XAMPP-CONTROL.EXE` steuern.

Weitere Infos finden Sie unter: <http://www.apachefriends.org/de/xampp.html>
Webserver für Mac OS X (ebenfalls auf der Buch-DVD):

<http://www.mamp.info/de/index.php>

Um die Einträge des Gästebuchs dauerhaft zu speichern, benutzen Sie in diesem Workshop eine MySQL-Datenbank.

Auf der DVD zum Buch finden Sie im zum Workshop gehörenden Projektordner die Datei `DB_CREATE_POSTS.SQL`, die für die Erstellung der nötigen Datenbanktabelle mit dem Namen `posts` sorgt.

Die Struktur der Tabelle sieht wie folgt aus:

Feldname	Datentyp
post_id	INT(10) UNSIGNED
post_name	VARCHAR(25)
post_www	VARCHAR(100)
post_message	TEXT
post_date	DATETIME

Sie haben somit für jedes Eingabefeld innerhalb des Flash-Films eine Spalte in der angelegten Datenbanktabelle. Hinzu kommt lediglich noch eine Spalte, in der das Erstellungsdatum gespeichert wird.

Wählen Sie also eine Datenbank auf Ihrem Webserver aus, bzw. erstellen Sie eine neue Datenbank mit einem Namen Ihrer Wahl. Anschließend führen Sie das mitgelieferte SQL-Skript so aus, dass die Tabelle in dieser Datenbank erstellt wird.

Datenbanktabelle erstellen

Um auf Ihrem Webserver mit Hilfe von phpMyAdmin eine Datenbanktabelle zu erstellen, öffnen Sie zuerst die phpMyAdmin-Oberfläche in Ihrem Webbrowser. Jetzt wählen Sie eine bereits vorhandene Datenbank aus der Spalte am linken Rand der Seite aus oder erzeugen über das Eingabefeld in der Mitte der Seite eine neue Datenbank mit einem Namen Ihrer Wahl. Wichtig ist dabei zu beachten, dass die gewählte Kollation `utf8` ist – so können die Inhalte der Datenbank ohne weitere Konvertierung in Ihrem Flash-Film angezeigt werden.

Nachdem Sie die Datenbank ausgewählt bzw. erstellt haben, öffnen Sie über die SQL-Schaltfläche unterhalb des phpMyAdmin-Logos in der linken oberen Ecke des Browserfensters das SQL-Eingabe-Fenster. Hier können Sie über den Reiter `DATEIMPORT` die auf der Buch-DVD mitgelieferte SQL-Datei hochladen. Nachdem die SQL-Anweisung erfolgreich ausgeführt wurde, erhalten Sie eine Bestätigung und die neu erstellte Datenbanktabelle sollte in der linken Spalte des Browserfensters auftauchen.

Schritt 12

Datenbanktabelle anlegen

◀ Tabelle 1

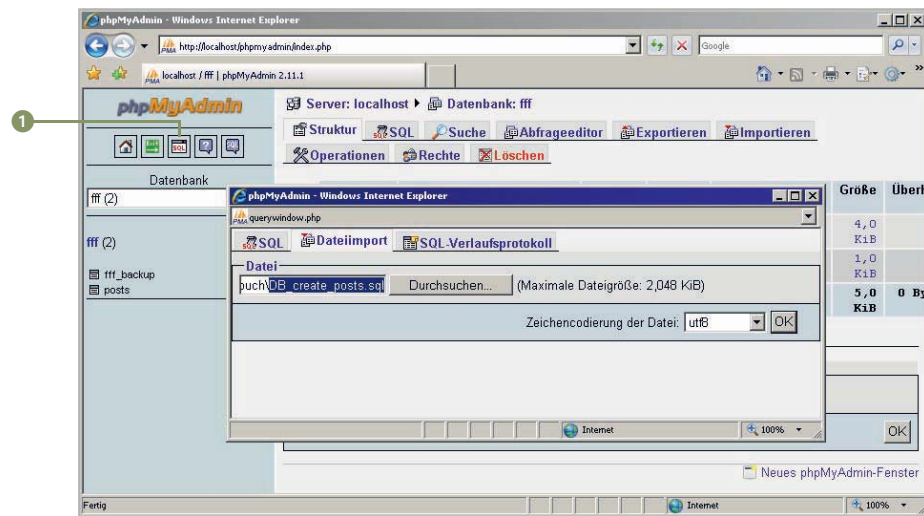
Datenstruktur der Tabelle »posts«. In der linken Spalte finden Sie die Namen der Datenbankfelder, in der rechten Spalte den jeweiligen Datentyp.

Info

Zur Pflege bzw. zum Anlegen der Datenbanktabellen bietet sich ein Admin-Tool wie z. B. **phpMyAdmin** mit einer grafischen Oberfläche an.

Sollten Sie den XAMPP-Webserver installiert haben, können Sie die mitgelieferte Version von phpMyAdmin benutzen, um lokal die benötigte Datenbank zu erstellen. <http://www.phpmyadmin.net>

Abbildung 10 ►
Klicken Sie auf den SQL-Button ❶, um das SQL-Skript auszuführen.



Schritt 13 Remoting-Service schreiben

Da Sie nun den Grundstein der Gästebuch-Anwendung gelegt haben, können Sie sich daran machen, Ihren ersten Remoting-Service zu schreiben.

Ein Service ist nichts anderes als eine PHP-Klasse, die Sie auf den Webserver kopieren. Das Gateway ermöglicht Ihnen dann, direkt aus dem ActionScript heraus, Funktionen des Services auf dem Server aufzurufen. Diese Funktionen können z. B. auf die eben erstellte Datenbank zugreifen und Ihnen die bereits erstellten Einträge zurückliefern.

Aber ein Schritt nach dem anderen! Zunächst benötigen Sie einen solchen Service. Hierzu öffnen Sie einen Texteditor Ihrer Wahl und erstellen ein neues Textdokument.

Das Grundgerüst des Services sieht folgendermaßen aus:

```
1: <?php
2: class Post
3: {
4:     var $conn;
5:     var $db_host = "server_url";
6:     var $db_user = "username";
7:     var $db_password = "passwort";
8:     var $database = "datenbankname";
9: }
10: ?>
```

Speichern Sie diese Datei als POST.PHP in dem Unterordner SERVICES des bereits angelegten Ordners AMFPHP. Der Pfad zu der Datei sollte demnach wie folgt aussehen: WEBROOT/AMFPHP/SERVICES/POST.PHP

Die Werte der Datenbankvariablen aus den Zeilen 5 bis 8 müssen Sie durch Ihre Zugangsdaten ersetzen.

Jetzt fehlt Ihnen lediglich noch eine Methode innerhalb des Services, die Ihnen die Gästebuch-Einträge aus der Datenbank ausliest.

Zum Extrahieren der Einträge fügen Sie folgende `get`-Methode direkt unterhalb der letzten Instanzvariablen der Datei `POST.PHP` hinzu:

```
1: public function get()
2: {
3:     $answer = array();
4:     $this->conn = mysql_connect($this->db_host,$this->
        db_user,$this->db_password);
5:     mysql_select_db($this->database,$this->conn);
6:     $sql = "SELECT * FROM posts ORDER BY post_date DESC";
7:     $result = mysql_query($sql);
8:     if(mysql_num_rows($result) != FALSE)
9:     {
10:         $answer["status"] = "ok";
11:         $answer["posts"] = array();
12:         while($row = mysql_fetch_array($result))
13:         {
14:             . . .
15:         }
16:     }
17:     else $answer["status"] = "no_posts_available";
18:     mysql_close($this->conn);
19:     return $answer;
20: }
```

In dem `$answer`-Array in Zeile 3 werden Sie das Ergebnis, in diesem Fall die aus der Datenbank ausgelesenen Gästebucheinträge, speichern. Dieses Array werden Sie am Ende der Methode an den Flash-Film zurückgeben. Zeile 4 stellt über die PHP-interne `mysql_connect`-Funktion die Verbindung mit der Datenbank her und speichert diese als Referenz in der `$conn`-Variablen. Die Funktion erwartet als Übergabeparameter die URL zum Datenbankserver, den Benutzernamen und das Passwort für den Zugriff auf die Datenbank. Daraufhin können Sie in Zeile 5 mit der `mysql_select_db`-Methode und dem Datenbanknamen sowie der Referenz (der zuvor erstellten Verbindung) die Datenbank auswählen.

Schritt 14

Methode zum Auslesen der Einträge definieren

Verbindung schliessen

Das Schließen der Verbindung ist optional, da nicht persistente Verbindungen automatisch bei Beendigung des PHP-Skripts geschlossen werden. Es soll Ihnen aber verdeutlichen, dass die Datenbankverbindung nach jeder Anfrage wieder abbricht.

In Zeile 6 folgt der benötigte SQL-Befehl zum Auslesen der Datenbank: Sie wünschen alle Felder (*) der `posts`-Tabelle absteigend geordnet (`ORDER BY ... DESC`) nach dem Datum (`post_date`). Zeile 7 startet die Abfrage und speichert das zurückgelieferte Ergebnis in der temporären Variablen `$result`.

Die `if`-Anweisung in Zeile 8 überprüft mit Hilfe der `mysql_num_rows`-Funktion, ob überhaupt Einträge gefunden wurden. Wenn dies der Fall ist, setzen Sie in Zeile 10 den Status-Parameter der Antwort auf `ok`. Anschließend erzeugen Sie innerhalb des `$answer`-Arrays ein weiteres Array, das die ausgelesenen Einträge beinhalten wird. Über die `while`-Schleife in Zeile 12 holen Sie sich das Ergebnis mit der `mysql_fetch_array`-Funktion Eintrag für Eintrag und speichern es in der `$row`-Variablen. In Zeile 14 würden Sie diese Werte in das dafür vorgesehene Array schreiben. Da dieses Vorgehen aber einer längeren Erklärung bedarf, lassen Sie Zeile 14 vorerst leer.

Sollten keine Einträge gefunden worden sein, wird die `else`-Anweisung in Zeile 17 ausgeführt und der Status-Parameter auf `no_posts_available` (dt.: keine Einträge vorhanden) gesetzt.

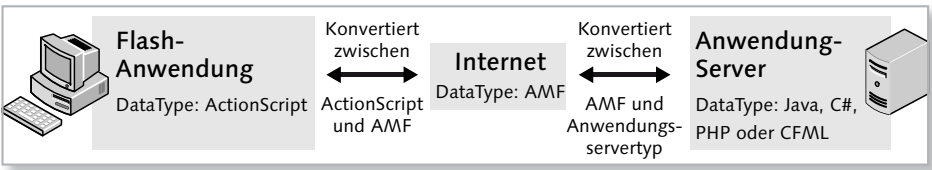
Bevor die Methode das Ergebnis in Zeile 19 an den Flash-Film zurückgibt, wird in Zeile 18 die Verbindung zur Datenbank wieder geschlossen.

Schritt 15 Rückgabedaten formatieren

Flash Remoting ermöglicht die direkte Kommunikation zwischen dem Flash-Frontend und dem jeweiligen Backend – in Ihrem Fall dem AMFPHP-Gateway. Alle nativen Datentypen können zwischen den beiden Stationen hin- und hergeschickt werden, wobei sich das Gateway um die automatische Konvertierung der Datentypen kümmert.

Das Gästebuch in diesem Beispiel arbeitet aber nicht mit nativen Datentypen, sondern eigens hierfür erstellten Objekten, den Gästebucheinträgen. AMFPHP ermöglicht Ihnen, mit Hilfe sogenannter **Value Objects** den Austausch eigener Datentypen zu realisieren. Sie erzeugen dazu auf beiden Seiten des Gateways jeweils eine Klasse, die alle Eigenschaften eines Gästebucheintrags besitzt. Danach werden diese beiden Klassen über die Konfiguration des Gateways aufeinander abgebildet (engl.: mapped), so dass Objekte dieser Klassen von dem einen in das andere konvertiert werden können und umgekehrt.

Abbildung 11 ►
Kommunikation von Flash
und PHP



Als Erstes erzeugen Sie das Value Object als ActionScript-Klasse. Wechseln Sie hierzu zurück zu Flash, und legen Sie über `Strg/⌘+N` eine neue ACTIONSCRIPT-DATEI an. Fügen Sie die folgende Klasse in das leere Dokument ein:

```
1: package
2: {
3:     public class PostVO
4:     {
5:         public var id:uint;
6:         public var name:String;
7:         public var www:String;
8:         public var message:String;
9:         public var time:int;
10:    }
11: }
```

Die Klasse verfügt über keinen Konstruktor, sie definiert lediglich einige für einen Gästebucheintrag spezifische Eigenschaften.

Speichern Sie diese Klasse mit `Strg/⌘+S` als `PostVO.as` in demselben Ordner wie die bereits bestehende `MAIN.as`.

Nun erstellen Sie das entsprechende Gegenstück auf Seiten des Gateways – die `PostVO.php`. Öffnen Sie ein weiteres leeres Dokument in Ihrem Texteditor, und fügen Sie die folgenden Zeilen hinzu:

```
1: <?php
2: class PostVO
3: {
4:     var $id;
5:     var $name;
6:     var $www;
7:     var $message;
8:     var $time;
9:
10:    public function PostVO($obj=null)
11:    {
12:        if($obj != null)
13:        {
14:            $this->id = $obj['post_id'];
15:            $this->name = $obj['post_name'];
16:            $this->www = $obj['post_www'];
17:            $this->message = $obj['post_message'];
18:            $this->time = strtotime($obj['post_date']);
```

Schritt 16

Value Objects für das Gästebuch anlegen

```

19:         }
20:     }
21: }
22: ?>

```

Achten Sie darauf, dass die Eigenschaften in beiden Klassen identische Namen tragen.

Die PHP-Klasse ähnelt der ActionScript-Klasse bis auf die Ausnahme, dass diese Klasse doch einen Konstruktor besitzt. Dieser dient der schnellen Erstellung eines PostVO-Objekts auf Seiten von PHP. Auf diese Weise können Sie dem Konstruktor ein assoziatives Array übergeben, das direkt die Werte der einzelnen Eigenschaften des Objekts setzt.

Speichern Sie die Änderungen als `PostVO.php` in demselben Ordner wie die `Post.php` auf Ihrem Webserver ab.

Schritt 17

Gateway-Konfiguration anpassen

Damit die beiden erstellten Klassen »gemapped« werden können, öffnen Sie die Datei `ADVANCEDSETTINGS.php` im `AMFPHP`-Ordner auf Ihrem Webserver und fügen die nachfolgenden zwei Zeilen hinzu.

Das `$incoming`-Array erweitern Sie um:

```
'PostVO' => 'PostVO'
```

Im `$outgoing`-Array kommt hinzu:

```
'postvo' => 'PostVO'
```

Die erste Änderung »mapped« die eingehenden PostVO-ActionScript-Objekte auf die entsprechende PHP-Klasse, die zweite die PHP-Objekte auf die ActionScript-Klasse. Die angepassten Zeilen sollten nun wie folgt aussehen:

```

19: $incoming = array(
20: ...
29:     'PostVO' => 'PostVO'
30: );
31: ...
35: $outgoing = array(
36: ...
44:     'postvo' => 'PostVO'
45: );

```

Mit dieser Änderung können Sie nun PostVO-Objekte zwischen dem ActionScript und PHP hin- und hertransferieren.

Speichern Sie die Datei ab, und schließen Sie sie dann.

Wechseln Sie erneut zur Datei Post.PHP in den Texteditor.

Damit Sie auf die erstellte PostVO-Klasse zugreifen können, müssen Sie diese zuerst importieren. Dies geschieht über die `include_once`-Funktion in Zeile 2 der geöffneten Datei:

```
include_once("PostVO.php");
```

Jetzt können Sie sich um die Verarbeitung der aus der Datenbank zurückgelieferten Einträge kümmern. Die im Schritt 14 ausgelassene Implementierung in Zeile 14 sieht mit Hilfe des VO wie folgt aus:

```
array_push($answer["posts"], new PostVO($row));
```

Hier erzeugen Sie ein neues PostVO-Objekt, übergeben diesem die aus der Datenbank ausgelesenen Werte und fügen das erstellte Objekt dann dem bereits bestehenden Array hinzu.

Je nachdem wie viele Einträge in der Datenbank stehen, liefert die Methode ein Array zurück, das den Status der Anfrage sowie alle Einträge innerhalb eines weiteren Arrays beinhaltet.

Beispielhaft könnte ein Ergebnis folgendermaßen aussehen:

```
array(
    'status': 'ok',
    'posts': array($post1, $post2)
);
```

In diesem Schritt werden Sie das erste Mal auf das Remoting-Gateway zugreifen und die bereits bestehenden Einträge im Gästebuch abrufen. Öffnen Sie hierzu die MAIN.AS, und erweitern Sie diese um die folgenden Zeilen.

Als Erstes benötigen Sie zwei neue externe Klassen. Fügen Sie sie unter den bisherigen Import-Anweisungen ein:

```
import flash.net.Responder;
import flash.net.registerClassAlias;
```

Schritt 18

Service-Methode vervollständigen

Schritt 19

Bestehende Beiträge abrufen

registerClassAlias()

Die `registerClassAlias`-Funktion sorgt dafür, dass ein Objekt dieser bestimmten Klasse nach der Konvertierung in das Action Message Format (AMF) den gleichen Typ aufweist wie zuvor. So stellen Sie sicher, dass die zwischen dem Remoting-Gateway und der Flash-Anwendung hin- und hergeschickten Objekte immer den vorher ausgewählten, richtigen Typ besitzen.

Der `Responder` wird die Ergebnisse des Remoting-Aufrufs verarbeiten; und die `registerClassAlias`-Funktion sorgt dafür, dass die entsprechenden Objekte beim Konvertieren ins Action Message Format (kurz: AMF) typsicher umgewandelt werden.

Sie wollen gleich am Anfang der Anwendung die bereits bestehenden Einträge abrufen und anzeigen. Zu diesem Zweck fügen Sie als erste Zeile in den Konstruktor das Mapping der `PostVO`-Klasse wie folgt ein:

```
registerClassAlias("PostVO", PostVO);
```

Fügen Sie dann unterhalb des Konstruktors die noch fehlenden Methoden hinzu:

```
1: private function getPosts():void
2: {
3:     modal_mc.visible = true;
4:     var resp:Responder = new Responder(getPostsOk, onFault);
5:     conn.call("Post.get", resp);
6: }
7: private function getPostsOk(res:Object):void
8: {
9:     modal_mc.visible = false;
10:    if(res.status == "ok") drawPosts(res.posts);
11:    else error_txt.text = res.status;
12: }
13: private function onFault(error:Object):void
14: {
15:     modal_mc.visible = false;
16:     error_txt.text = error.description;
17: }
```

Die `getPosts`-Methode in Zeile 1 sorgt dafür, dass die Anfrage an das Remoting-Gateway geschickt wird. Hierzu benutzen Sie in Zeile 5 das `NetConnection`-Objekt und rufen damit die Service-Funktion `Post.get` auf dem Server auf. Als zweiten Parameter übergeben Sie der `call`-Methode den in der vorherigen Zeile erzeugten `Responder`. Der Konstruktor der `Responder`-Klasse erwartet zum einen die Methode, die aufgerufen werden soll, wenn das Ergebnis erfolgreich zurückgeliefert wurde (`getPostsOk()`); zum anderen eine Methode, die bei einem Fehler des Aufrufs ausgeführt werden soll (`onFault()`).

Bevor Sie den Aufruf abschicken, blenden Sie in Zeile 3 den modalen Layer ein, so dass das Gästebuch für die Zeit der Kommunikation zwischen Frontend und Backend inaktiv geschaltet wird.

In Zeile 7 folgt die `getPostsOk`-Methode, die die Erfolgsantwort des Gateways verarbeitet. Da der Service-Aufruf beendet ist, blendet die Methode in Zeile 9 als Erstes den modalen Layer wieder aus. Danach wird mit einer `if`-Anweisung überprüft, ob die Antwort des Services Gästebucheinträge beinhaltet (`res.status == "ok"`) oder das Gästebuch bisher leer ist. Sollten Einträge vorhanden sein, werden diese in Zeile 10 an die `drawPosts`-Methode übergeben, die deren weitere Darstellung übernimmt. In dem Fall, dass noch keine Einträge im Gästebuch stehen, wird dies in Zeile 11 in das `error_txt`-Textfeld auf der Bühne geschrieben. Das entspricht zwar nicht unbedingt einem Fehler, doch um dem Besucher der Webseite Feedback zu geben, schreiben Sie den Rückgabewert in das erstellte Textfeld.

Wenn ein Fehler beim Service-Aufruf auftritt, wird die in Zeile 13 definierte `onFault`-Methode aufgerufen. Diese blendet den modalen Layer aus und schreibt die zurückgelieferte Fehlermeldung in das `error_txt`-Textfeld. In diesem Fall wird die Fehlermeldung ausgegeben, die aus dem Backend an den Flash-Film übergeben wurde. Idealerweise würden Sie diese in eine für den Besucher verständliche Meldung umschreiben.

Die im vorherigen Schritt angesprochene `drawPosts`-Methode wurde bisher noch nicht implementiert – dies werden Sie nun nachholen. Vorher benötigen Sie jedoch noch zwei Helfer-Klassen.

Als Erstes legen Sie über `[Strg]/[⌘]+[N]` eine neue ACTIONSCRIPT-DATEI an. Das wird unsere Post-Klasse, die die Darstellung der einzelnen Einträge übernimmt. Das Grundgerüst der Klasse sieht wie folgt aus:

```
1: package
2: {
3:     import flash.display.Sprite;
4:     import flash.text.TextField;
5:     import flash.text.TextFieldAutoSize;
6:     import flash.text.TextFormat;
7:     public class Post extends Sprite
8:     {
9:         public function Post(postData:PostV0)
10:        {
11:        }
12:    }
13: }
```

Schritt 20

Post-Klasse für die Darstellung der Gästebucheinträge

Die Imports in den Zeilen 3 bis 6 erklären sich so: Da Sie die einzelnen Post-Objekte später auf der Bühne platzieren wollen, diese aber keine Zeitleiste benötigt, erbt die Post-Klasse nicht von `MovieClip`, sondern von `Sprite`. Die anderen drei externen Klassen werden zum Erstellen und Bearbeiten von Textfeldern benötigt.

Der Konstruktor in Zeile 9 erwartet als Übergabeparameter ein `PostVO`-Objekt. Dieses liefert alle nötigen Informationen für die Darstellung des Eintrags.

Schritt 21

Post-Textfelder anlegen

Fügen Sie nun diese Zeilen innerhalb des Konstruktors ein:

```
1: var format:TextFormat = new TextFormat("Arial",10,0x666666);
2: var title_txt:TextField = new TextField();
3: title_txt.x = 9;
4: title_txt.y = 0;
5: title_txt.autoSize = TextFieldAutoSize.LEFT;
6: title_txt.defaultTextFormat = format;
7: var message_txt:TextField = new TextField();
8: message_txt.x = 10;
9: message_txt.y = 15;
10: message_txt.width = 432;
11: message_txt.wordWrap = true;
12: message_txt.autoSize = TextFieldAutoSize.LEFT;
13: format.size = 11;
14: format.bold = true;
15: message_txt.defaultTextFormat = format;
16: addChild(title_txt);
17: addChild(message_txt);
```

Zeile 1 erzeugt ein `Textformat`, das Sie weiter unten in der Methode auf die beiden noch zu erstellenden Textfelder anwenden werden. Der erste Parameter steht hierbei für die verwendete Schrift – in diesem Fall `Arial` –, der zweite Parameter legt die Größe der Schrift in Pixel fest. Als Letztes übergeben Sie die Farbe der Schrift als hexadezimalen Wert: `0x666666` entspricht dabei einem mittleren Grau.

In Zeile 2 erzeugen Sie das erste von zwei Textfeldern `title_txt`, das später die Informationen über den Autor und den Zeitpunkt des Eintrags beinhalten wird.

Die beiden darauffolgenden Zeilen positionieren das Textfeld. Zeile 5 setzt `autoSize` auf `true`, was bewirkt, dass das Textfeld in seiner Breite an

den Inhalt angepasst wird. Anschließend legen Sie das erzeugte Textformat als `defaultTextFormat` fest.

Das zweite Textfeld, das als Container für die eigentliche Nachricht bestimmt ist, erstellen Sie in Zeile 7. Neben der Positionierung in den Zeilen 8 und 9 und der Änderung der Breite in Zeile 10 setzen Sie in den folgenden Zeilen die Parameter `wordWrap` und `autoSize`. Das hat den Effekt, da dieses Textfeld mehrzeilig ist, dass der Text am Ende jeder Zeile umbricht und sich das Textfeld automatisch an die Höhe seines Inhalts anpasst.

Paul schrieb am 22.9.2007 um 17:30 Uhr

At vero eos et accusamus et iusto odio dignissimos ducimus, qui blanditiis praesentium voluptatum deleniti atque corrupti, quos dolores et quas molestias excepturi sint, obcaecati cupiditate non provident, similique sunt in culpa, qui officia deserunt mollitia animi, id est laborum et dolorum fuga.

◀ Abbildung 12

So soll ein Gästebucheintrag im fertigen Gästebuch angezeigt werden.

Bevor Sie in Zeile 15 das Textformat des zweiten Textfeldes festlegen, passen Sie das vorhandene Format an. Zeile 13 vergrößert die Schrift um ein Pixel, und Zeile 14 lässt die Schrift `bold` erscheinen.

Schließlich fügen Sie die beiden erstellten Textfelder in den Zeilen 16 und 17 mit der `addChild`-Methode der Anzeigeliste hinzu.

Das beim Erstellen eines Eintrags übergebene `PostVO`-Objekt beinhaltet alle notwendigen Informationen. Diese müssen Sie lediglich noch in die richtige Form bringen, bevor Sie sie anzeigen können.

Die folgenden Zeilen sorgen für die nötige Formatierung – fügen Sie sie oberhalb der beiden `addChild`-Anweisungen ein:

```
1: var date:Date = new Date(postData.time*1000);
2: var dateStr:String = date.getDate()+"."+date.getMonth()+1)+"."+date.getFullYear();
3: var minuteStr:String = (date.getMinutes() < 10) ? "0"+date.getMinutes() : date.getMinutes().toString();
4: var timeStr:String = date.getHours()+":"+minuteStr;
5: var authorStr:String;
6: if(postData.www == "") authorStr = postData.name;
7: else authorStr = "<a href='http://'+postData.www+' ' target='_blank'><font color='#66CCFF'>" + postData.name + "</font></a>";
8: title_txt.htmlText = authorStr + " schrieb am " + dateStr + " um " + timeStr + " Uhr";
9: message_txt.text = postData.message;
```

Schritt 22

Eintragsdaten formatieren

Da Sie den Erstellungszeitpunkt des Eintrags vom Backend in Sekunden erhalten, ActionScript intern jedoch mit Millisekunden rechnet, multiplizieren Sie den Wert in Zeile 1 mit 1000. Diese Zahl benutzen Sie zur Erstellung eines neuen `Date`-Objekts. Ausgehend von diesem Objekt, initialisieren Sie in Zeile 2 einen String, der das Datum in folgendem Format speichert: TT.MM.JJJJ. Zusätzlich zum Datum wollen Sie den genauen Zeitpunkt angeben. In der nachfolgenden Zeile bestimmen Sie die Minuten und füllen diese, wenn nötig, mit einer führenden Null auf. Die führende Null bewirkt, dass die Stunden und Minuten immer als eine zweistellige Zahl angezeigt werden. Andernfalls würde es in manchen Fällen zu solchen Zeiten führen: »17:5« – obwohl es eigentlich »17:05« heißen müsste. Zeile 4 komplettiert den genauen Zeitpunkt: Hier werden die Stunden zusammen mit den Minuten in dem String `timeStr` in dem Format `SS:MM` hinterlegt.

Da nicht alle Angaben des Autors beim Erstellen eines Eintrags Pflichtfelder sind, überprüfen Sie in den Zeilen 5 bis 7 mit einer `if`-Anweisung, ob der Autor seine eigene Webadresse angegeben hat oder nicht. Sollte der Autor dieses Feld leer gelassen haben, geben Sie ihn lediglich mit seinem Namen an. Hat er allerdings eine Webseite angegeben, verlinken Sie in Zeile 7 die URL mit dem Namen des Autors.

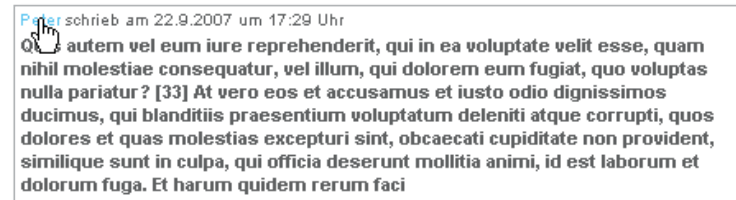


Abbildung 13 ►
Der Name ist mit der URL
des Autors verlinkt.

Anschließend setzen Sie in Zeile 8 den Inhalt des Textfeldes `title_txt`, in dem alle Angaben wie Zeitpunkt und Autor des Eintrags stehen. Die eigentliche Nachricht übernehmen Sie in Zeile 9 eins zu eins in das zweite Textfeld, `message_txt`.

Speichern Sie die erstellte Klasse mit `[Strg]/[⌘]+[S]` unter dem Namen `Post.As` ab.

Schritt 23

Die Spacer-Klasse anlegen

Als zweiten Helfer benötigen Sie die `Spacer`-Klasse. Ein solcher `Spacer` kommt jeweils vor den ersten und hinter den letzten Eintrag des Gästebuchs. So sorgen Sie dafür, dass oberhalb und unterhalb der Einträge ein gewisser Abstand eingehalten wird. Dies ist eine rein optische Veränderung, die die Lesbarkeit der Einträge verbessert.

Legen Sie über `Strg`/`⌘`+`N` eine weitere ACTIONSCRIPT-DATEI an, und füllen Sie diese mit folgenden Zeilen:

```
1: package
2: {
3:     import flash.display.Sprite;
4:     public class Spacer extends Sprite
5:     {
6:         public function Spacer(size:uint, yPos:int)
7:         {
8:             graphics.beginFill(0xFF0000, 0);
9:             graphics.drawRect(0,0,size,size);
10:            y = yPos;
11:        }
12:    }
13: }
```

Die Zeilen 1 bis 4 sollten Ihnen inzwischen bekannt vorkommen. Da die Klasse in keinem untergeordneten Package liegt, geben Sie in Zeile 1 keinen weiteren Package-Namen an. Zeile 3 importiert die benötigte Sprite-Klasse. In Zeile 4 erbt `Spacer` von der eben importierten Klasse `Sprite`.

Der Konstruktor in Zeile 6 erwartet als Übergabeparameter die Größe des Platzhalters in Pixel und in diesem Fall die aktuelle y-Position. Die Zeilen 8 und 9 innerhalb des Konstruktors sorgen dafür, dass ein transparentes Quadrat in der übergebenen Größe gezeichnet wird. Die nächste Zeile verschiebt das komplette Sprite auf die neue y-Position.

Speichern Sie die fertige `Spacer`-Klasse mit `Strg`/`⌘`+`S` unter dem Namen `SPACER.AS` im aktuellen Workshop-Ordner ab.

Nun können Sie sich an das eigentliche Anzeigen der Gästebucheinträge machen. Hierzu öffnen Sie erneut die `MAIN.AS` und fügen die noch fehlende Methode `drawPosts()` hinzu:

```
1: private function drawPosts(posts:Array):void
2: {
3:     while(posts_mc.numChildren > 0) posts_mc.removeChildAt(0);
4:     posts_mc.addChild(new Spacer(10,0));
5:     var currentY:int = 10;
6:     for(var i:int=0;i<posts.length;i++)
7:     {
8:         var tPost:Post = new Post(posts[i]);
```

Packages

Mehr über Packages erfahren Sie im Exkurs »Klassen, Vererbung, Packages«.

Schritt 24

Gästebucheinträge anzeigen

```

9:         tPost.y = currentY;
10:        posts_mc.addChild(tPost);
11:        currentY += tPost.height+10;
12:    }
13:    posts_mc.addChild(new Spacer(10, currentY));
14:    sp.source = posts_mc;
15: }

```

Diese Methode erwartet in Zeile 1 als einzigen Parameter ein Array mit den darzustellenden Einträgen.

Die `while`-Schleife in Zeile 3 sorgt dafür, dass alle vorherigen Einträge im `posts_mc`-MovieClip wieder von der Bühne entfernt werden – dies wird erst wichtig, wenn Sie die `drawPosts`-Methode mehr als ein Mal aufrufen. Die alten Beiträge müssen zuerst gelöscht werden, da Sie ansonsten die neuen Beiträge unterhalb der alten anfügen. Zudem tauchen die Beiträge dann doppelt im Gästebuch auf.

Zeile 4 fügt den ersten – im vorherigen Schritt besprochenen – Platzhalter ein. Danach setzen Sie die temporäre Positions-Variable `currentY` auf 10. So beginnen die Einträge erst ab einer y-Position von 10 Pixel.

Nun folgt die `for`-Schleife, die in Zeile 6 das komplette Einträge-Array durchläuft und für jeden Eintrag in Zeile 8 ein neues Post-Objekt erstellt. Diesem Objekt werden alle notwendigen Informationen (`posts[i]`) mit übergeben. Es liefert ein fertiges Sprite zurück, das in Zeile 9 positioniert und in Zeile 10 der Anzeigeliste hinzugefügt wird, so dass es auf der Bühne zu sehen ist. Anschließend wird in Zeile 11 die aktuelle y-Position um die Höhe des zuletzt erstellten Eintrags und einen zusätzlichen Abstand von 10 Pixel vergrößert.

Nachdem alle Post-Objekte erstellt wurden, fügen Sie in Zeile 13 den zweiten Platzhalter unterhalb des letzten Eintrags ein. Zeile 14 weist nun noch den `posts_mc` als `source` der ScrollPane zu. Dies sorgt dafür, dass die Einträge innerhalb der ScrollPane angezeigt werden.

Schritt 25

Testen des Flash-Films

Um die Änderungen zu testen, speichern Sie die `MAIN.AS` zunächst mit `Strg`/`⌘`+`S` ab. Anschließend können Sie das Ergebnis mit `Strg`/`⌘`+`↵` begutachten.

Wenn Sie das Backend auf Ihrem eigenen Server umgesetzt haben, dann sollten Sie noch keinen Eintrag in der Datenbank vorfinden. Es erscheint die Meldung »no_posts_available« ¹, und die ScrollPane auf der rechten Seite bleibt leer. Benutzen Sie hingegen den *flashfastforward.de*-Server, sollten zumindest einige Einträge zurückgeliefert und dargestellt werden.



▲ **Abbildung 14**

Die Einträge werden aus der Datenbank ausgelesen und korrekt angezeigt.

Bisher stellt der Service nur eine Methode zum Abholen bereits bestehender Einträge zur Verfügung. In diesem Schritt werden Sie die entsprechende Methode zum Erstellen von Einträgen innerhalb des Services anlegen.

Hierzu öffnen Sie die sich im `root`-Verzeichnis Ihres Webserver befindende Datei `Post.php` und ergänzen die Klasse um die folgende Methode:

```
1: public function add($post)
2: {
3:     $answer = array();
4:     $this->conn = mysql_connect($this->db_host,$this->
        db_user,$this->db_password);
5:     mysql_select_db($this->database,$this->conn);
6:     $p_name = mysql_escape_string($post->name);
7:     $p_www = mysql_escape_string($post->www);
8:     $p_message = mysql_escape_string($post->message);
9:     $sql = "INSERT INTO posts (post_name, post_www,
        post_message, post_date) VALUES('$p_name', '$p_www',
        '$p_message', NOW())";
10:    if(!mysql_query($sql)) $answer['status'] =
        "add_post_failed";
11:    else $answer['status'] = "ok";
12:    mysql_close($this->conn);
13:    return $answer;
14: }
```

Die `add`-Methode erwartet als Übergabeparameter ein `PostVO`-Objekt, das alle Informationen, die zur Erstellung eines Eintrags benötigt werden, beinhaltet.

Schritt 26

Service-Methode zum Erstellen von Einträgen anlegen

Achtung

Sollten Sie auf den *flashfast-forward.de*-Server zugreifen und die Backend-Implementierung nicht selbst übernehmen haben, überspringen Sie diesen Punkt und machen Sie direkt mit Schritt 27 weiter.

Am Anfang der Methode erstellen Sie in Zeile 3 wiederum ein leeres Array `$answer`, das im weiteren Verlauf das Ergebnis des Funktionsaufrufs speichern wird.

In den beiden nächsten Zeilen stellen Sie – genau wie in der `get`-Methode zuvor – die Verbindung zur Datenbank her. Anschließend bereiten Sie die einzelnen Werte des übergebenen Eintrags für das Schreiben in die Datenbank vor. Mit Hilfe der `mysql_escape_string`-Funktion werden mögliche Sonderzeichen so maskiert, dass sie ohne Probleme in der SQL-Anweisung verwendet werden können. In Zeile 9 definieren Sie die SQL-Anweisung, die die formatierten Werte für »Name des Autors«, »Webseite des Autors« und die eigentliche Nachricht in die entsprechenden Tabellenfelder schreibt. Zusätzlich setzen Sie über `NOW()` den Zeitpunkt der Erstellung des Eintrags. Die `if`-Anweisung in Zeile 10 führt zu einer die SQL-Anweisung aus, überprüft zum anderen aber gleichzeitig auch, ob diese einen Fehler zurückliefert. Sollte dies der Fall sein, wird der Status-Parameter des `$answer`-Arrays auf `add_post_failed` gesetzt. Wenn der Eintrag ohne Fehler hinzugefügt wurde, setzen Sie den Status in Zeile 11 auf `ok`.

Nun beenden Sie in Zeile 12 noch die Verbindung zur Datenbank und geben das komplette `$answer`-Array an Flash zurück.

Speichern Sie die vorgenommenen Änderungen des Service ab, und wechseln Sie dann zurück zu Flash.

Schritt 27

Gästebucheintrag erstellen

Ihre Anwendung beherrscht schon das Auslesen von Gästebucheinträgen – jetzt machen Sie sich daran, eigene Einträge zu erstellen. Die nötigen Formularfelder haben Sie bereits in Schritt 4 dieses Workshops angelegt. Es folgen nun das Auslesen und Abschicken der Daten an das Remoting-Gateway. Hierzu öffnen Sie erneut die Main-Klasse und erweitern diese unter den bisherigen Import-Anweisungen um folgende Zeilen:

```
import flash.events.MouseEvent;
```

Die Klasse `MouseEvent` benötigen Sie, damit Sie über der `NetConnection` aus Schritt 8 einen Klick auf den ABSENDEN-Button abfangen bzw. umleiten können:

```
send_btn.addEventListener(MouseEvent.CLICK, sendPost);
```

Mit diesem neuen EventListener sorgen Sie dafür, dass bei einem Klick auf den ABSENDEN-Button (`send_btn`) die `sendPost`-Methode ausgeführt wird.

Nun folgen die `sendPost`- und die `sendPostOk`-Methoden, die Sie ans Ende Ihrer Klasse einfügen:

```
1: private function sendPost(evt:MouseEvent):void
2: {
3:     modal_mc.visible = true;
4:     var tempPost:PostVO = new PostVO();
5:     tempPost.name = name_txt.text;
6:     tempPost.www = (www_txt.text == "") ? null : www_txt.text;
7:     tempPost.message = message_txt.text;
8:     var resp:Responder = new Responder(sendPostOk, onFault);
9:     conn.call("Post.add", resp, tempPost);
10: }
11: private function sendPostOk(res:Object):void
12: {
13:     modal_mc.visible = false;
14:     if(res.status == "ok")
15:     {
16:         name_txt.text = www_txt.text = message_txt.text = "";
17:         getPosts();
18:     }
19:     else
20:     {
21:         error_txt.text = res.status;
22:     }
23: }
```

Die `sendPost`-Methode in Zeile 1 erwartet als Übergabeparameter ein `MouseEvent`, da sie direkt durch einen Klick auf den ABSENDEN-Button aufgerufen wird.

Wie schon zuvor wird auch in dieser Methode während der ganzen Transaktion der modale Layer in Zeile 3 eingeblendet. Danach erzeugen Sie in Zeile 4 ein temporäres `PostVO`-Objekt, das in den drei darauffolgenden Zeilen die Werte der Eingabefelder aus dem Formular übernimmt. Dabei wird nur bei der optionalen Angabe der Webadresse darauf geachtet, dass kein leerer String, sondern stattdessen `null` übergeben wird. So ist es auf Seiten des Backends einfacher zu kontrollieren, ob ein Wert definitiv nicht gesetzt wurde.

Zeile 8 legt den benötigten `Responder` fest, der die weitere Abarbeitung bestimmt: Wenn der Eintrag erfolgreich gespeichert wurde, wird die `sendPostOk`-Methode aufgerufen, bei einem Fehler die `onFault`-Methode. In Zeile 9 wird die Anfrage ans Gateway geschickt. Neben dem Namen der

null

In der Informatik spricht man von `null`, wenn ein Wert nicht vorhanden bzw. noch nicht gesetzt ist. Dies darf nicht verwechselt werden mit der Zahl Null.

Service-Methode und dem Responder geben Sie diesmal einen weiteren Parameter mit: das temporäre PostVO-Objekt. Dies enthält alle notwendigen Informationen, so dass der Eintrag vom Backend in die Datenbank eingefügt werden kann.

Da Sie die `onFault`-Methode schon in Schritt 19 definiert haben, springen Sie nun direkt zur Implementierung der `sendPostOK-Callback-Methode`. Diese erwartet in Zeile 11 als Übergabeparameter das vom Service zurückgelieferte Ergebnis (engl.: result) als `Object`. Da die Verarbeitung zu Ende ist, blenden Sie in Zeile 13 den modalen Layer wieder aus. Danach überprüfen Sie in Zeile 14 den Status des übergebenen Ergebnisses. Ist dieser `ok`, setzen Sie zuerst alle drei Eingabefelder zurück und holen sich anschließend über den `getPosts`-Aufruf die neuen Gästebucheinträge aus der Datenbank. Sollte der Status nicht `ok` sein, ist ein Fehler beim Einfügen Ihres Eintrags passiert und Sie geben die Fehlerbeschreibung in Zeile 21 im `error_txt`-Textfeld auf der Bühne aus.

Schritt 28

Formulareingaben überprüfen

Bevor Sie die neue Funktion Ihres Gästebuchs das erste Mal testen, sollten Sie einen Check einbauen, der überprüft, ob auch wirklich alle erforderlichen Formularfelder ausgefüllt wurden. Wenn das nicht der Fall ist, haben Sie unvollständige Einträge in der Datenbank. Um dies zu vermeiden, fügen Sie als erste Zeile in der `sendPost`-Methode diese `if`-Anweisung hinzu:

```
if(!checkInputs()) return;
```

Innerhalb der erstellten `if`-Anweisung wird die Methode `checkInputs()` aufgerufen, die die Formularfelder überprüft. Sollte hierbei eine Unstimmigkeit auftauchen, liefert diese Methode `false` zurück und die weitere Ausführung der `sendPost`-Methode bricht mit dem `return`-Befehl ab.

Hier nun die Implementierung der `checkInputs`-Methode am Ende der Klasse:

```
1: private function checkInputs():Boolean
2: {
3:     if(name_txt.text.length < 1)
4:     {
5:         error_txt.text = "Bitte geben Sie einen Namen ein.";
6:         name_txt.borderColor = 0xff0000;
7:         stage.focus = name_txt;
8:         return false;
```

```

9:     }
10:    if(message_txt.text.length < 1)
11:    {
12:        error_txt.text = "Bitte geben Sie eine Nachricht
13:        ein.";
14:        message_txt.borderColor = 0xff0000;
15:        stage.focus = message_txt;
16:        return false;
17:    }
18:    if(www_txt.text.length != 0 && www_txt.text.length < 10)
19:    {
20:        error_txt.text = "Bitte geben Sie eine korrekte URL
21:        ein.";
22:        www_txt.borderColor = 0xff0000;
23:        stage.focus = www_txt;
24:        return false;
25:    }
26:    return true;
27: }

```

Die `checkInputs`-Methode überprüft mit drei `if`-Anweisungen, ob die jeweiligen Formularfelder richtig befüllt wurden.

In Zeile 3 kontrollieren Sie die Länge des in `name_txt` stehenden Textes. Ist diese kleiner als 1, so ist das Feld leer. Aus diesem Grund geben Sie in Zeile 5 einen Fehler aus und heben in Zeile 6 die falsche Eingabe durch die rote Einfärbung des Textfeld-Rahmens hervor. Mit `stage.focus` in Zeile 7 setzen Sie den Cursor in das hervorgehobene Formularfeld. In Zeile 8 geben Sie wegen des Fehlers dann noch `false` an die aufrufende Methode zurück.

Die Kontrolle der eingegebenen Nachricht in den Zeilen 10 bis 16 läuft genauso ab. Wenn die Länge des Textes nicht stimmt, wird ein Fehler ausgegeben.

Lediglich bei der Überprüfung des optionalen Eingabefelds »Webseite« müssen Sie weiter unterscheiden. Die `if`-Anweisung in Zeile 17 stellt zuerst sicher, dass die Länge des eingegebenen Textes nicht 0 ist. Zudem muss, wenn etwas eingegeben wurde, der Text mindestens zehn Zeichen lang sein (z.B. »www.ard.de«). Wenn diese Voraussetzungen nicht erfüllt sind, geben Sie auch hier einen Fehler aus – genau wie bei den beiden Eingabefeldern zuvor.

Tritt kein Fehler auf, teilen Sie dies der `sendPost`-Methode mit, indem Sie in Zeile 24 `true` zurückgeben.

Info

In der Regel werden Formulardaten mit Hilfe von **Regulären Ausdrücken** (kurz: REGEX) überprüft. Wenn Sie mehr dazu erfahren wollen, finden Sie auf der Website zum Buch unter www.galileodesign.de/1537 einen Workshop, der sich ausschließlich mit diesem Thema befasst.

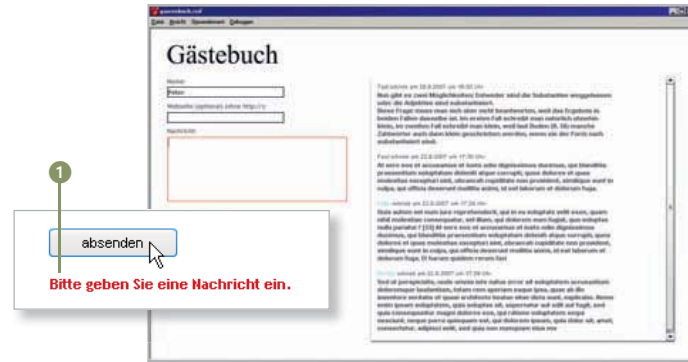
Schritt 29

Testen des Flash-Films

Speichern Sie Ihre Änderungen an der MAIN.AS mit **[Strg]/[⌘]+[S]** ab. Die neu hinzugefügten Funktionen können Sie nun über **[Strg]/[⌘]+[↵]** testen.

Versuchen Sie zuerst, einen Eintrag zu erstellen, ohne dass Sie alle Felder ausgefüllt haben. Es sollte die von Ihnen definierte Fehlermeldung erscheinen.

Abbildung 15 ▶
Das Textfeld für die Nachricht wurde nicht ausgefüllt und wird deshalb rot umrandet. Außerdem wird eine Fehlermeldung ❶ ausgegeben.



Jetzt korrigieren Sie die Eingabe und senden das Formular erneut ab. Wenn alles klappt, sollte Ihr Eintrag im Gästebuch auftauchen.

Abbildung 16 ▶
Der Gästebucheintrag wurde erstellt ❷, jedoch wird immer noch die Fehlermeldung angezeigt.



Aber leider sehen Sie auch, dass noch immer die Fehlermeldung erscheint und das Eingabefeld weiterhin hervorgehoben ist. Dies werden Sie im nächsten Schritt beheben.

Schritt 30

Fehlermeldung zurücksetzen

Um die einmal angezeigte Fehlermeldung wieder zurückzusetzen, öffnen Sie ein letztes Mal die MAIN.AS und fügen diese Zeile zu den Import-Anweisungen hinzu:

```
import flash.events.Event;
```

Damit importieren Sie die Event-Klasse, die Sie in den nun folgenden Zeilen benötigen. Fügen Sie im Konstruktor oberhalb der erzeugten `NetConnection` folgende Zeilen hinzu:

```
name_txt.addEventListener(Event.CHANGE, onChange);
www_txt.addEventListener(Event.CHANGE, onChange);
message_txt.addEventListener(Event.CHANGE, onChange);
```

Mit diesen Zeilen fügen Sie an jedes der drei Eingabefelder einen Event-Listener an, der auf Veränderungen des Textfeldinhalts reagiert. Wenn dies geschieht, wird die `onChange`-Methode aufgerufen. Ergänzen Sie die Methode am Ende der Klasse:

```
1: private function onChange(evt:Event):void
2: {
3:     error_txt.text = "";
4:     name_txt.borderColor = 0x000000;
5:     www_txt.borderColor = 0x000000;
6:     message_txt.borderColor = 0x000000;
7: }
```

Die `onChange`-Methode sorgt zum einen in Zeile 3 dafür, dass die Fehlermeldung zurückgesetzt wird. Zum anderen ändert sie in den Zeilen 4 bis 6 die Farbe der drei Textfeldrahmen wieder auf das ursprüngliche Schwarz.

Speichern Sie zum letzten Mal die `MAIN.AS` mit `[Strg]/[⌘]+[S]` ab. Wechseln Sie zur `GAESTEBUCH.FLA`, und veröffentlichen Sie Ihr fertiges Gästebuch final über `[⬆]+[F12]`.

Schritt 31

Veröffentlichen des Flash-Films



◀ **Abbildung 17**
Das Gästebuch ist fertig – Gratulation!

Audio und Video

Audio-Player.....	342
Soundspektrum	370
Adobe AIR.....	377
AIR-Musikplayer	380
Video-Player	387

Audio-Player

Ein eigener MP3-Player in Flash CS3

```
package
{
    import flash.display.MovieClip;
    import flash.display.SimpleButton;
    import flash.text.TextField;
    import flash.media.Sound;
    import flash.media.SoundChannel;
    import flash.media.SoundLoaderContext;
    import flash.media.SoundTransform;
    import flash.net.URLRequest;
    import flash.events.Event;
    import flash.events.ProgressEvent;
    import flash.events.MouseEvent;
    import flash.geom.Rectangle;
    import flash.media.SoundTransform;
    public class Player extends MovieClip
    {
        private const BUFFER_TIME:int = 3000;
        private var songs:Array = ["song01.mp3","song02.mp3"];
```

Wollten Sie schon immer einmal Musiktitel in Ihre Flash-Seite einbinden? Oder vielleicht Ihre persönliche Playlist den Besuchern Ihrer Seite zugänglich machen? Dies alles werden Sie in diesem Workshop lernen. Sie bauen Ihren eigenen Audio-Player, der externe MP3-Dateien abspielt und dem Benutzer alle Freiheiten einer richtigen Anwendung gibt.

Zielsetzungen:

- Player-Grafiken erstellen
- MP3-Datei abspielen
- Bedienelemente mit Funktionen belegen

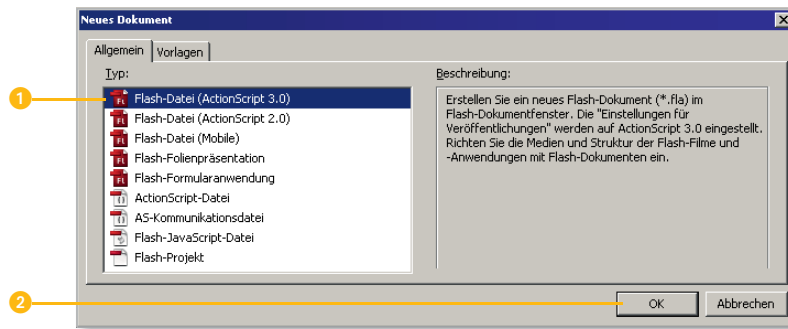
[Ergebnis: 30_AUDIO-PLAYER

• MUSICPLAYER1.SWF]



Im ersten Schritt legen Sie einen neuen Ordner auf Ihrer Festplatte an und nennen diesen MUSICPLAYER1. Hier werden Sie die von Ihnen erstellten Beispiele zu diesem Workshop abspeichern.

Starten Sie die Flash-Autorenumgebung, und erstellen Sie über DATEI • NEU... eine neue FLASH-DATEI (ACTIONSCRIPT 3.0) ❶. Bestätigen Sie mit OK ❷. Speichern Sie anschließend die Datei direkt im Projektordner ab.



Schritt 1

Vorbereitungen

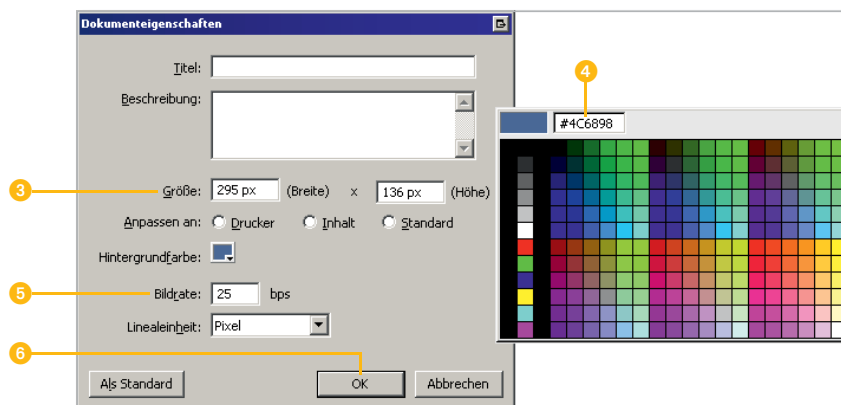
Schritt 2

Flash-Datei erzeugen

◀ Abbildung 1

Flash-Datei MUSICPLAYER1.FLA anlegen

Da der Audio-Player klein und kompakt werden soll, öffnen Sie mit `[Strg]/[⌘] + [J]` das Fenster DOKUMENTEIGENSCHAFTEN und passen die Werte wie folgt an: Die Grösse ändern Sie in 295x136 Pixel ❸, die HINTERGRUNDFARBE bekommt einen Wert von #4C6898 ❹, und die BILDRATE setzen Sie auf 25 Bilder pro Sekunde (bps) ❺. Schließen Sie das Fenster mit OK ❻.



Schritt 3

Dokumenteigenschaften anpassen

◀ Abbildung 2

Die Dokumenteigenschaften für den MP3-Player festlegen

Sie fangen mit dem Erstellen der einfachen Hintergrundgrafik des Players an. Diese Grafik sollte so groß sein, dass später alle nötigen Player-Elemente darauf Platz finden. Im Beispiel zu diesem Workshop füllt die Hin-

Schritt 4

Hintergrundgrafik erstellen

Grafiken gestalten

In diesem Workshop wird die Erstellung diverser Grafiken beispielhaft erklärt. Sie können entweder auf die Grafiken aus der Beispieldatei der Buch-DVD zurückgreifen oder auch eigene Grafiken erzeugen bzw. in Flash importieren. Lassen Sie Ihrer Kreativität freien Lauf!

tergrundgrafik fast die komplette Bühne aus und besteht aus übereinanderliegenden Vektorflächen, wobei die hintere der beiden leicht transparent ist. Verteilen Sie die beiden Grafiken am besten auf unterschiedlichen Ebenen, so kommen sie sich beim Zeichnen nicht in die Quere.

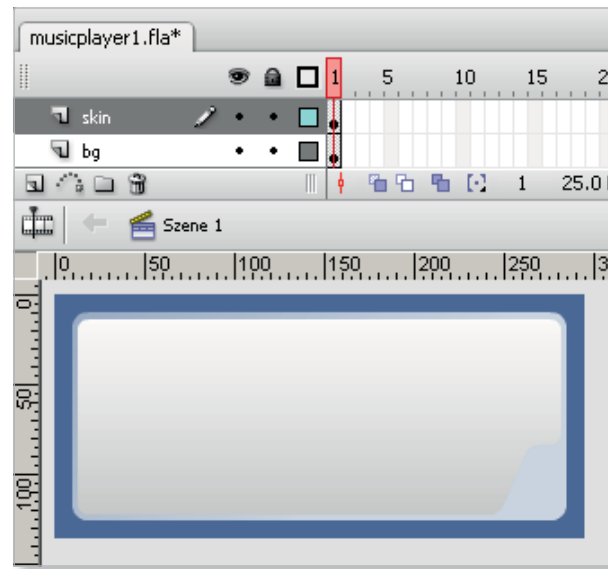


Abbildung 3 ►

Beim Zeichnen des Audio-Players sind Ihnen gestalterisch keine Grenzen gesetzt.

Schritt 5 Display hinzufügen

Jeder Audio-Player verfügt über eine Anzeige (engl.: display), auf der die aktuellen Informationen wie z. B. Musiktitel angezeigt werden. In der Beispieldatei wird die verwendete Display-Grafik DISPLAY.PNG über **Strg**/**⌘** + **R** aus dem Ordner ASSETS auf die Bühne importiert und auf einer weiteren Ebene positioniert.

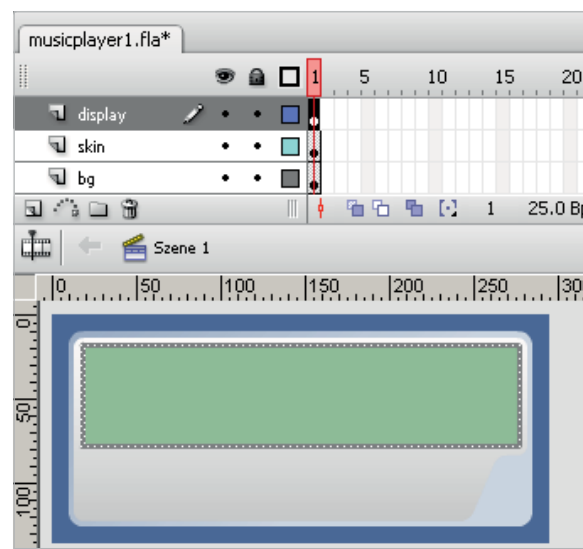
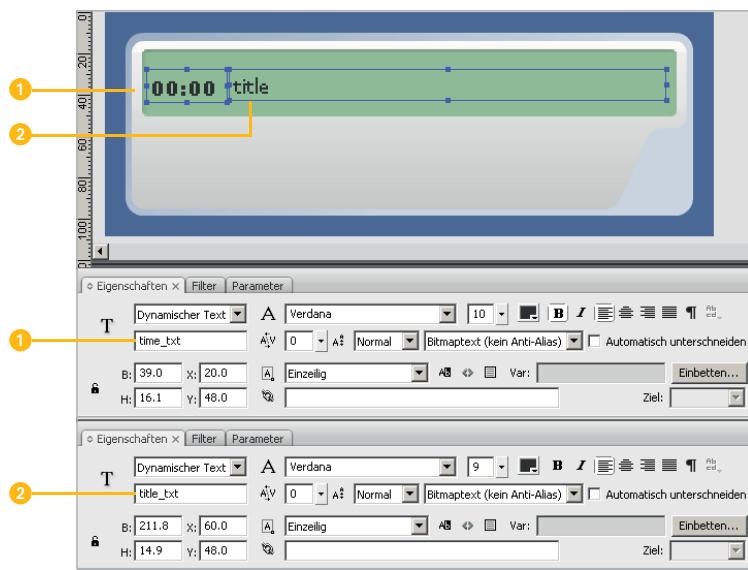


Abbildung 4 ►

Die Grafik für das Display wird platziert.

Die Elemente, die auf dem Display erscheinen, sind unter anderem die momentane Abspielzeit des Musikstücks, sein Titel und die aktuelle Abspiel- bzw. Ladeposition. Die Zeit und den Titel werden Sie in jeweils einem dynamischen Textfeld ausgeben; die Position visualisieren Sie mit Hilfe eines Statusbalkens.

Auf einer weiteren Ebene »content« erstellen Sie nun zwei dynamische Textfelder: »time_txt« ❶ und »title_txt« ❷.



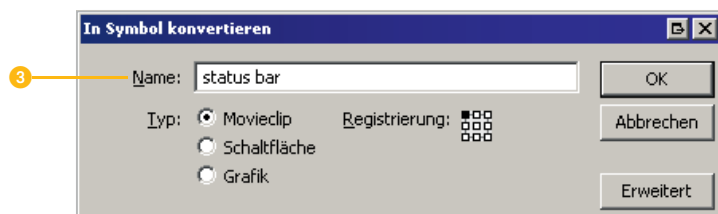
Schritt 6

Display erweitern

◀ **Abbildung 5**
Zwei dynamische Textfelder nehmen den Titel und die Abspielzeit des Titels auf.

Unterhalb der eben erstellten Textfelder legen Sie den Statusbalken an. Zeichnen Sie hierzu ein 249 x 7 Pixel großes Rechteck, und konvertieren es über [F8] in ein Symbol.

Benennen Sie den Movieclip im IN SYMBOL KONVERTIEREN-Dialog mit »status bar« ❸, und bestätigen Sie die Konvertierung mit OK.



Schritt 7

Statusbalken anlegen

◀ **Abbildung 6**
Der Ladebalken wird in einen Movieclip umgewandelt.

Das so erstellte Symbol wählen Sie auf der Bühne aus, kopieren es mit [Strg]/[⌘] + [C] und fügen es ein zweites Mal auf genau derselben Position mit [Strg]/[⌘] + [⇧] + [V] wieder ein. Dem oberen der beiden Movieclips geben Sie über den FARBE-Effekt im Eigenschaftensfenster die Farbe #4A5C4A ❹.

und verringern anschließend die Breite auf 120 Pixel **2**. Der untere Balken erhält die Farbe #7B997B **5**. Als Instanznamen wählen Sie für den ersten Balken »positionBar_mc« **1** und für den zweiten »loadingBar_mc« **3**.

Wie Sie den Namen bereits entnehmen können, wird der erste Balken für die Positionsanzeige und der zweite für die Ladestatusanzeige zuständig sein.

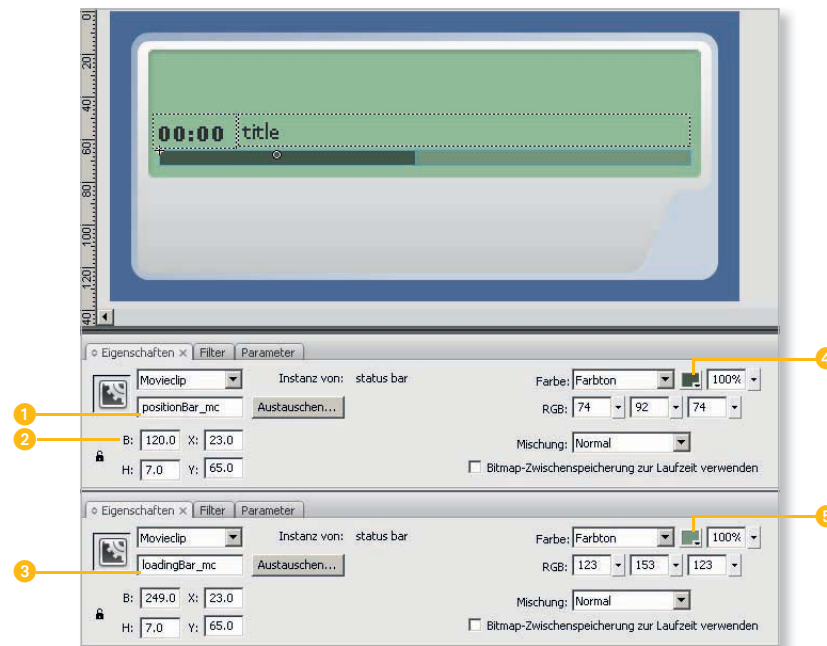


Abbildung 7 ▶

Ein Balken mit zwei Funktionen: Der untere Balken zeigt den Ladestatus an, der obere gibt die Abspielposition innerhalb des Liedes wieder.

Schritt 8 Steuerungselemente ergänzen

Jetzt fehlen noch die nötigen Buttons zum Bedienen des Audio-Players. Erstellen Sie jeweils einen Button für Abspielen, Pausieren, Stoppen, Zurück- und Vorwärtsspringen.

Legen Sie über EINFÜGEN • ZEITLEISTE • EBENE eine weitere Ebene an, und zeichnen Sie auf dieser ein 26x21 Pixel großes Viereck. Konvertieren Sie das Viereck mit **F8** in eine SCHALTFLÄCHE **7**, und geben Sie ihr den Namen »btn play« **6**.

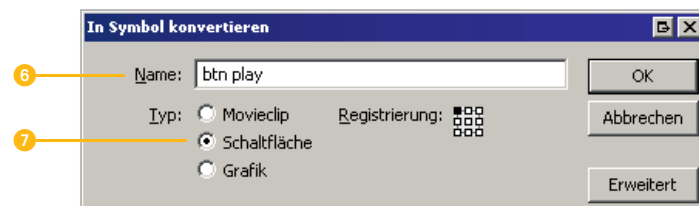
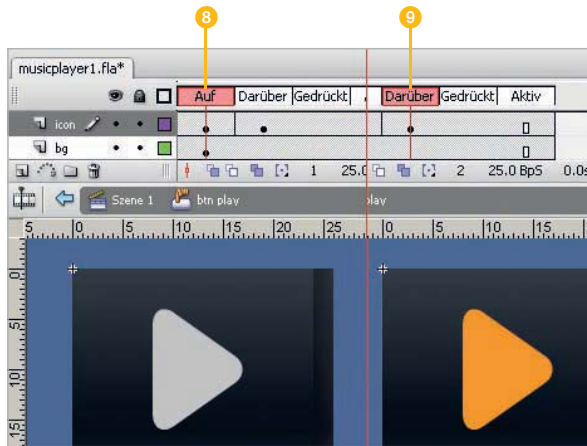


Abbildung 8 ▶

Der Play-Button wird in ein Symbol konvertiert.

Nachdem Sie das Symbol erstellt haben, wechseln Sie mit einem Doppelklick auf das Viereck auf der Bühne in den Symbol-Bearbeitungsmodus.

Bisher ist der Abspielen-Button noch nicht als solcher zu erkennen. Damit die Steuerelemente am Ende leicht voneinander zu unterscheiden sind, verwenden Sie am besten die gleichen Zeichen, die auf jedem DVD- oder CD-Player zu finden sind. Für den Play-Button also ein nach rechts zeigendes graues Dreieck ⁸. Im DARÜBER-Zustand erstellen Sie ein neues Schlüsselbild und färben das Dreieck orange ein ⁹.

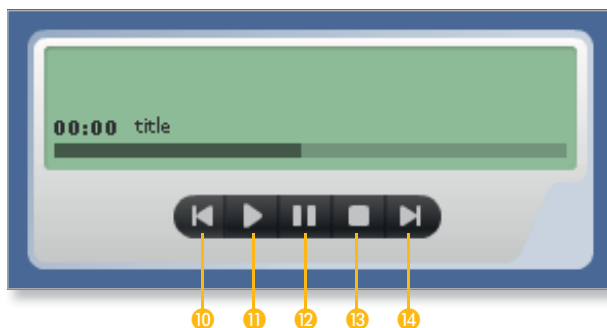


◀ **Abbildung 9**

AUF- und DARÜBER-Zustand des Play-Buttons

Die noch fehlenden vier Schaltflächen legen Sie nach genau diesem Muster an. In der Beispieldatei zu diesem Workshop sehen die fertigen Steuerelemente wie folgt aus. Von links nach rechts:

- ¹⁰ Zurück
- ¹¹ Abspielen
- ¹² Pause
- ¹³ Stopp
- ¹⁴ Vorwärts



◀ **Abbildung 10**

Die fertigen Steuerelemente

Geben Sie den Schaltflächen in der oben genannten Reihenfolge die folgenden Instanznamen:

- ▶ »prev_btn«
- ▶ »play_btn«
- ▶ »pause_btn«
- ▶ »stop_btn«
- ▶ »next_btn«

Schritt 9 Anlegen der Dokumentklasse

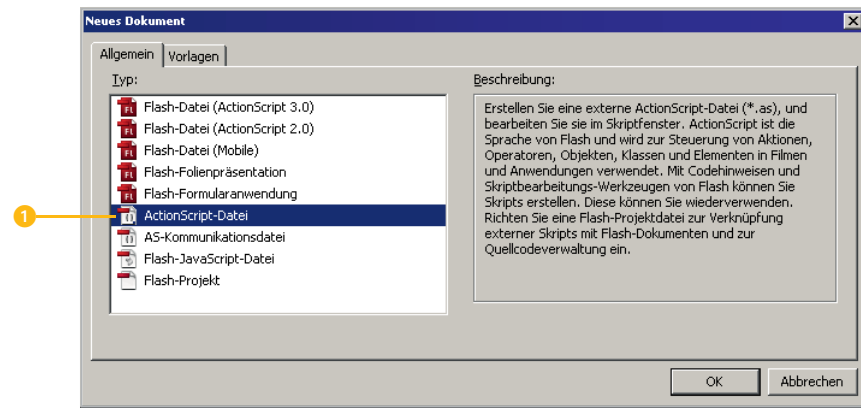
Zwischenergebnis:
30_AUDIO-PLAYER •
PLAYER_STEP09.AS



Abbildung 11 ►

Neue ActionScript-Datei für
die Dokumentklasse

Bisher haben Sie lediglich die Vorbereitungen für den Audio-Player vorgenommen – jetzt werden Sie dem Ganzen Leben einhauchen. Zu diesem Zweck legen Sie über `Strg/⌘ + N` eine neue ACTIONSCRIPT-DATEI an 1.



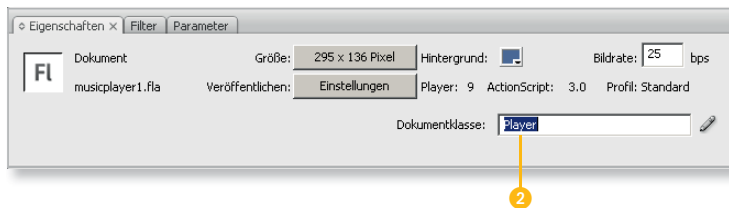
In der neu angelegten Datei werden Sie in den nächsten Schritten dieses Workshops alle Funktionen des Players umsetzen. Fangen Sie mit dem grundlegenden Klassengerüst an:

```
1: package
2: {
3:     import flash.display.MovieClip;
4:     import flash.display.SimpleButton;
5:     import flash.text.TextField;
6:     public class Player extends MovieClip
7:     {
8:         public function Player()
9:         {
10:        }
11:    }
12: }
```

Jede Dokumentklasse erbt in der Regel entweder von `flash.display.MovieClip` oder `flash.display.Sprite`. Damit wird sichergestellt, dass alle

notwendigen Voraussetzungen geschaffen sind. In diesem Fall wählen Sie **MovieClip** und importieren die benötigte Klasse in Zeile 3. Weil auch bereits Schaltflächen und Textfelder auf der Bühne liegen, stellen Sie in den darauffolgenden Zeilen zusätzlich die **SimpleButton**- und die **TextField**-Klasse per Import zur Verfügung.

Speichern Sie die Klasse anschließend unter dem Namen **PLAYER.AS** im Workshop-Ordner ab. Wechseln Sie zurück zu **MUSICPLAYER1.FLA**, und weisen Sie der Datei innerhalb des Eigenschaftensfensters »Player« als **DOKUMENTKLASSE** zu ❶. Bestätigen Sie mit .



Kopieren Sie nun zwei MP3-Dateien in den Workshop-Ordner, und benennen Sie diese in »song01.mp3« und »song02.mp3« um.

Öffnen Sie erneut die Datei **PLAYER.AS**, und ergänzen Sie folgende Codezeilen unter den bisherigen Import-Anweisungen:

```
import flash.media.Sound;
import flash.media.SoundChannel;
import flash.media.SoundLoaderContext;
import flash.media.SoundTransform;
import flash.net.URLRequest;
```

Diese fünf Klassen werden zum Laden und anschließenden Abspielen eines Sounds benötigt. Die wichtigsten Funktionen werden im weiteren Verlauf des Workshops erklärt – mehr über die jeweilige Klasse können Sie der Flash-Hilfe entnehmen.

Definieren bzw. initialisieren Sie nun die folgenden Variablen innerhalb der **Player**-Klasse direkt über dem Konstruktor:

```
1: private const BUFFER_TIME:int = 3000;
2: private var songs:Array = ["song01.mp3", "song02.mp3"];
3: private var currentIndex:int = 0;
4: private var currentSound:Sound;
```

MP3-Files auf der Buch-DVD

Wenn Sie gerade keine MP3-Dateien zur Hand haben, können Sie die Dateien von der Buch-DVD nutzen. Beachten Sie jedoch, dass diese ausschließlich für Ihre Übungen, und nicht kommerziell eingesetzt werden dürfen.

◀ Abbildung 12

Die Dokumentklasse wird im Eigenschaftensfenster der Flash-Datei zugewiesen.

Schritt 10

Imports organisieren



Zwischenergebnis:
30_AUDIO-PLAYER •
PLAYER_STEP10.AS

Schritt 11

Erste Klassenvariablen



Zwischenergebnis:
30_AUDIO-PLAYER •
PLAYER_STEP11.AS

```

5: private var currentChannel:SoundChannel;
6: private var sBuffer:SoundLoaderContext = new
   SoundLoaderContext(BUFFER_TIME);
7: private var currentVolume:Number = 1;
8: private var isPlaying:Boolean = false;

```

In Zeile 1 legen Sie einen sogenannten Puffer (`BUFFER_TIME`) fest, der dafür sorgt, dass das Abspielen möglichst ohne Ladeunterbrechungen gelingt. Der Puffer ist in diesem Fall eine Konstante und beträgt 3000 Millisekunden oder einfacher: drei Sekunden. Demzufolge werden immer erst drei Sekunden des Songs geladen, bevor das Abspielen beginnt. Das Array in der nächsten Zeile ist Ihre Titelauswahl, die alle Songs beinhaltet, die abgespielt werden können. Der `currentIndex` in Zeile 3 sagt aus, welches der Stücke aus der Liste gerade zu hören ist. In den nächsten drei Zeilen wird jeweils eine `Sound`-, `SoundChannel`- und `SoundLoaderContext`-Variable deklariert. Die Funktion dieser Variablen werden Sie im Laufe des Workshops genauer verstehen. Zeile 7 wird die aktuelle Lautstärke speichern; und in Zeile 8 legen Sie ein Flag `isPlaying` an, das sich als boolesche Variable merkt, ob der Player momentan abspielt (`true`) oder nicht (`false`).

Schritt 12

Audio-Player initialisieren

Zwischenergebnis:
30_AUDIO-PLAYER •
PLAYER_STEP12.AS



Aus dem Konstruktor heraus werden Sie nun den Player initialisieren, d. h. zum einen alle nötigen Variablen setzen und Objekte erzeugen und zum anderen den ersten Song abspielen.

Erweitern Sie den Konstruktor von `PLAYER.AS` um diese Zeilen:

```

resetCurrentSong();
playSound(currentIndex);

```

Anschließend erweitern Sie die Klasse um die noch fehlenden Methoden – fügen Sie diese direkt unterhalb des Konstruktors ein.

```

1: private function resetCurrentSong():void
2: {
3:     title_txt.text = "";
4:     currentSound = new Sound();
5: }
6: private function playSound(index:int):void
7: {
8:     currentSound.load(new URLRequest(songs[index]), sBuffer);
9:     currentChannel = currentSound.play();

```



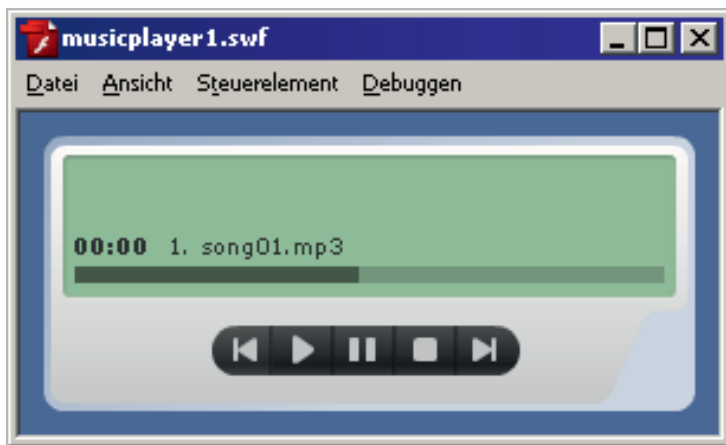
```

10:     isPlaying = true;
11:     title_txt.text = (currentIndex+1)+". " + songs[index];
12: }

```

Aus dem Konstruktor heraus rufen Sie die Methode `resetCurrentSong()` auf, die in Zeile 3 als Erstes das Textfeld für den Song-Titel leert und danach eine weitere Methode zum Abspielen des ersten Songs anstößt. Diese `playSound`-Methode beginnt in Zeile 6 und erwartet als Übergabeparameter eine Ganzzahl. In diesem Fall ist das der `currentIndex`, der die aktuelle Position in Ihrer Playlist darstellt. Anhand des Index und des bereits weiter oben festgelegten Puffers nutzen Sie in Zeile 8 die `load`-Funktion des Sound-Objekts dazu, die im Array hinterlegte Datei mit Hilfe eines `URLRequest` zu laden. In der darauffolgenden Zeile weisen Sie den geladenen Sound der `currentChannel`-Variablen vom Typ `SoundChannel` zu, die es Ihnen ermöglicht, den Sound im Nachhinein zu steuern. Weil der Song ab diesem Zeitpunkt abgespielt wird, setzen Sie in Zeile 10 die `isPlaying`-Variable auf `true`. Zusätzlich zeigen Sie mit dem Befehl in Zeile 11 die Position und den Dateinamen des aktuellen Songs in der Anzeige des Players an.

Speichern Sie die vorgenommenen Änderungen in `PLAYER.AS`, und testen Sie den Film mit `[Strg]/[F5]+[↩]`. Sie sollten den Song hören und den Titel des Songs in der Anzeige ablesen können.



Sicherlich ist Ihnen bereits aufgefallen, dass weder die Zeit hochzählt, noch der Statusbalken sich bewegt. Diese Funktionen werden Sie als Nächstes umsetzen. Im selben Schritt werden Sie die ID3-Tags Ihrer MP3-Dateien

Schritt 13

Flash-Film testen

◀ Abbildung 13

Der Song wird abgespielt und ist zu hören – vorausgesetzt natürlich, Sie haben Ihre Boxen eingeschaltet.

ID3-Tag

Wenn der von Ihnen über die `Sound.load`-Methode geladene Sound Metadaten in Form von ID3-Tags besitzt, können Sie diese Eigenschaften innerhalb Ihrer Flash-Anwendung abfragen.

auslesen und den Künstler- sowie Songnamen auf der Anzeige darstellen. Denn wer kann schon etwas mit »song01.mp3« anfangen?

Schließen Sie den Flash Player wieder, und widmen Sie sich erneut der Player-Klasse.

Schritt 14

Klassen-Imports und -variablen erweitern

Zwischenergebnis:
30_AUDIO-PLAYER •
PLAYER_STEP14.AS



Um die im vorherigen Schritt beschriebenen Eigenschaften umsetzen zu können, müssen Sie wiederum auf externe Klassen zugreifen. Diese importieren Sie wie folgt:

```
import flash.events.Event;  
import flash.events.ProgressEvent;
```

Zusätzlich benötigen Sie vier neue Klassenvariablen, die Sie unter den in Schritt 11 hinzugefügten deklarieren:

```
1: private const BAR_WIDTH:int = 249;  
2: private var sPercentLoaded:Number;  
3: private var sLength:int;  
4: private var sPosition:int;
```

In der Konstante in Zeile 1 hinterlegen Sie die maximale Breite des Statusbalkens, so dass Ihnen diese innerhalb der Klasse immer zur Verfügung steht. In der `sPercentLoaded`-Variable wird der jeweilige Ladestatus der MP3-Datei gespeichert. Die Variablen aus Zeile 3 und 4 benötigen Sie zur Bestimmung des Abspielstatus.

Schritt 15

Ladestatus und ID3-Daten anzeigen

Zwischenergebnis:
30_AUDIO-PLAYER •
PLAYER_STEP15.AS



Jetzt erweitern Sie die `resetCurrentSong`-Methode um die folgenden Zeilen – fügen Sie diese hinter den beiden bereits bestehenden Zeilen ein:

```
currentSound.addEventListener(Event.ID3, onID3);  
currentSound.addEventListener(ProgressEvent.PROGRESS,  
    onProgress);
```

In diesen beiden Zeilen fügen Sie dem Sound-Objekt jeweils einen neuen Ereignis-Listener hinzu. Der erste der beiden Listener ruft die `onID3`-Methode auf, sobald die ID3-Metadaten der geladenen MP3-Datei zur Verfügung stehen. Der zweite Listener ruft in gleichbleibenden Intervallen während des Ladens der Datei die `onProgress`-Methode auf. Diese beiden

Methoden werden Sie nun mit in die Klasse integrieren. Ergänzen Sie sie am Ende der Klasse:

```
1: private function onID3(evt:Event):void
2: {
3:     title_txt.text = (currentIndex+1)+". " + currentSound.id3.
        artist + " - " + currentSound.id3.songName;
4: }
5: private function onProgress(evt:ProgressEvent):void
6: {
7:     sPercentLoaded = evt.bytesLoaded/evt.bytesTotal;
8:     loadingBar_mc.width = BAR_WIDTH*sPercentLoaded;
9: }
```

Innerhalb der `onID3`-Methode setzen Sie in Zeile 3 den Titel in der Anzeige des Players neu: Aus dem Dateinamen wird nun »Künstler – Songname«. Diese Informationen erhalten Sie aus der ID3-Eigenschaft des `currentSound`-Objekts.

Der `onProgress`-Methode wird ein `ProgressEvent` mit übergeben, das die bereits geladenen Bytes und die Gesamtzahl der Bytes der MP3-Datei mitliefert.

Anhand dieser beiden Werte berechnen Sie in Zeile 7 den prozentualen Wert der bereits geladenen Daten und zwischenspeichern diesen in der `sPercentLoaded`-Variablen. Anschließend aktualisieren Sie den Status (die Breite) des Ladebalkens.

In diesem Schritt aktualisieren Sie die Anzeige des Players in jedem neuen Bild (engl.: frame). Hierzu fügen Sie am Anfang der `playSound`-Methode folgenden Aufruf hinzu:

```
if(!hasEventListener(Event.ENTER_FRAME))
{
    addEventListener(Event.ENTER_FRAME, onEnterFrame);
}
```

Die `if`-Anweisung überprüft, ob bereits ein `ENTER_FRAME`-EventListener vorhanden ist. Sollte das nicht der Fall sein, wird dieser angelegt. Solch ein Listener bewirkt, dass in jedem neuen Bild, in diesem Fall also 25 Mal pro Sekunde, die `onEnterFrame`-Methode ausgeführt wird – die aber momentan noch fehlt.

Erweitern Sie die Player-Klasse deshalb ganz unten um folgende Zeilen.

ID3-Tags – weitere Eigenschaften

Sie fragen in diesem Beispiel nur zwei von vielen ID3-Eigenschaften ab. Eine komplette Liste aller Eigenschaften können Sie der Flash-Hilfe entnehmen.

Schritt 16

Anzeige stetig aktualisieren



Zwischenergebnis:
30_AUDIO-PLAYER •
PLAYER_STEP16.AS

```

1: private function onEnterFrame(evt:Event):void
2: {
3:     sLength = currentSound.length;
4:     sPosition = currentChannel.position;
5:     if(sPercentLoaded > 0)
6:     {
7:         sLength /= sPercentLoaded;
8:         positionBar_mc.width = BAR_WIDTH*sPosition/sLength;
9:         updateTime(sPosition);
10:    }
11: }
12: private function updateTime(pos:int):void
13: {
14:     var cMinutes:int = Math.floor(pos/1000/60);
15:     var cSeconds:int = Math.floor(pos/1000)%60;
16:     var mString:String = (cMinutes < 10) ? "0"+cMinutes :
        cMinutes.toString();
17:     var sString:String = (cSeconds < 10) ? "0"+cSeconds :
        cSeconds.toString();
18:     time_txt.text = mString + ":" + sString;
19: }

```

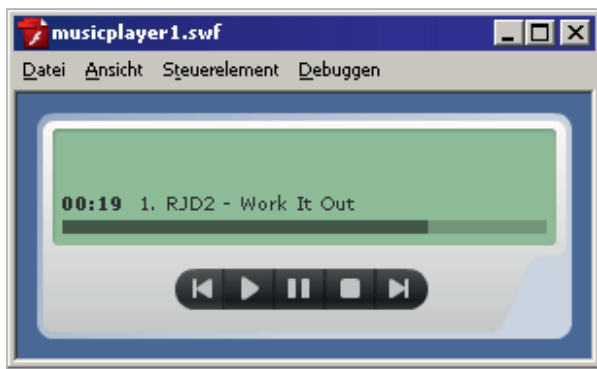
In Zeile 3 wird die momentane Länge des geladenen Sounds in der Variablen `sLength` zwischengespeichert. Hierbei gilt es, zu beachten, dass es nicht von Anfang an die komplette Länge des Songs ist, sondern lediglich die des bereits geladenen Teils. Die Position, die in der darauffolgenden Zeile als `sPosition` gespeichert wird, ist dagegen die absolute Position innerhalb des kompletten Songs – auch wenn dieser noch nicht vollständig geladen sein sollte. Mit der `if`-Anweisung in Zeile 5 überprüfen Sie, ob der Ladevorgang bereits begonnen hat – ob `sPercentLoaded` also größer als 0 ist. Ist dies der Fall, berechnen Sie in Zeile 7 aus der momentanen Länge und der prozentualen Gesamtlänge der geladenen Datei die komplette Länge in Millisekunden.

In Zeile 8 nutzen Sie die eben gewonnenen Informationen, um über die prozentuale Position innerhalb des Songs (`sPosition/sLength`) die Breite des Positionsbalkens zu bestimmen. Danach übergeben Sie den Wert der `sPosition`-Variablen an die `updateTime`-Methode, die in Zeile 14 aus den Millisekunden zuerst die Minuten errechnet und anschließend in Zeile 15 die verbleibenden Sekunden bestimmt. Die Zeilen 16 und 17 formatieren die Minuten und Sekunden zu einem String mit – wenn nötig – führenden Nullen, so dass alle Angaben in jedem Fall zwei Stellen haben. Abschließend setzen Sie in Zeile 18 den zusammengeführten Wert aus Mi-

nuten und Sekunden in das `time_txt`-Textfeld in der Anzeige des Players ein.

Speichern Sie die Änderungen in `PLAYER.AS` mit `Strg/⌘+S` ab.

Testen Sie die Neuerungen mit `Strg/⌘+↵`. Sie sollten den Namen des Künstlers und des Songs in der Anzeige sehen, ebenso die fortlaufende Zeit und die aktuelle Position anhand des Statusbalkens. Was Sie leider nicht sehen können, ist den Ladestatus. Da Sie die MP3-Datei lokal von Ihrem Rechner laden, gibt es in diesem Fall keine Verzögerung und die Daten stehen Ihnen sofort komplett zur Verfügung.



▲ **Abbildung 14**

Künstlernamen, Titel und fortlaufende Zeit werden angezeigt. Auch der Statusbalken funktioniert.

Momentan bricht der Player die Wiedergabe nach dem ersten Song ab. Um an diesem Punkt zum zweiten Song zu wechseln, benötigen Sie einen weiteren `EventListener` – in diesem Fall den `SOUND_COMPLETE`-Listener. Fügen Sie diesen innerhalb der `playSound`-Methode, nachdem Sie den Sound mit `currentChannel = currentSound.play();` starten, ein:

```
currentChannel.addEventListener(Event.SOUND_COMPLETE,  
    onSoundComplete);
```

Wie Sie sicherlich erkennen, soll bei Eintreten des `SOUND_COMPLETE`-Ereignisses die Methode `onSoundComplete()` aufgerufen werden. Diese fehlt bis zu diesem Zeitpunkt allerdings noch.

Fügen Sie Ihrer `Player`-Klasse ganz unten die folgenden drei Methoden hinzu.

Schritt 17

Flash-Film testen

Ladestatus testen

Wenn Sie den Ladestatus testen wollen, können Sie dies ohne großen Aufwand tun. Ersetzen Sie lediglich innerhalb der Datei `PLAYER.AS` die relativen Pfade zu den MP3-Dateien durch die folgenden absoluten Pfade:

```
[ "http://  
flashfastforward.de/  
sounds/song01.mp3",  
  "http://  
flashfastforward.de/  
sounds/song02.mp3" ]
```

Schritt 18

Abspielen aller Songs



Zwischenergebnis:
30_AUDIO-PLAYER •
PLAYER_STEP18.AS

```

1: private function onSoundComplete(evt:Event):void
2: {
3:     playNext();
4: }
5: private function playNext():void
6: {
7:     stopSound();
8:     currentIndex = (currentIndex < songs.length-1) ?
        ++currentIndex : 0;
9:     playSound(currentIndex);
10: }
11: private function stopSound():void
12: {
13:     removeEventListener(Event.ENTER_FRAME, onEnterFrame);
14:     currentChannel.stop();
15:     resetCurrentSong();
16:     isPlaying = false;
17: }

```

Wenn die `onSoundComplete`-Methode aufgerufen wird, leitet sie den Aufruf in Zeile 3, ohne diesen weiter zu bearbeiten, an die `playNext`-Methode weiter. Diese kümmert sich um den reibungslosen Wechsel zwischen den Songs. D.h., zuerst stoppt sie in Zeile 7 den aktuellen Song, setzt dann in Zeile 8 den Index um 1 hoch (bzw. springt zurück auf den ersten Song, wenn kein weiterer mehr vorhanden ist) und spielt den neuen Song über die bereits bestehende `playSound`-Methode ab.

In der `stopSound`-Methode ab Zeile 11 wird als Erstes der `ENTER_FRAME`-EventListener entfernt, danach in Zeile 14 das Abspielen des Songs angehalten und dann mit `resetCurrentSound()` in Zeile 15 ein neues Sound-Objekt erzeugt. Schließlich setzen Sie in Zeile 16 die `isPlaying`-Variable auf `false`, da zu diesem Zeitpunkt der neue Sound nicht abgespielt wird.

Schritt 19

Flash-Film testen

Speichern Sie die Änderungen mit `Strg/⌘+S` in `PLAYER.AS`, und testen Sie dann den Flash-Film über den Tastaturbefehl `Strg/⌘+↵`. Optisch gibt es keine Veränderungen, aber der Player sollte nun nicht nach dem ersten Song aufhören zu spielen, sondern zum zweiten Song springen und automatisch das Abspielen beginnen sowie die Anzeige aktualisieren.



◀ **Abbildung 15**

Auch der zweite Song wird abgespielt.

Momentan steuert der Player das Abspielen der Songs, ohne dass der Benutzer die Möglichkeit hat einzugreifen. Die Schaltflächen zur Steuerung des Players haben Sie bereits auf der Bühne angelegt und für die weitere Bearbeitung vorbereitet, indem Sie ihnen Instanznamen gegeben haben. Jetzt machen Sie sich daran, den Steuerelementen Leben einzuhauchen. Damit Sie die Maus-Interaktionen des Benutzers abfangen können, benötigen Sie eine weitere externe Klasse. Ergänzen Sie folgende Import-Anweisung:

```
import flash.events.MouseEvent;
```

Mit der Implementierung des Pause-Buttons entsteht ein weiterer Zustand des Players, den Sie über die folgende Variable wiedergeben. Fügen Sie die Zeile über dem Konstruktor ein:

```
private var isPausing:Boolean = false;
```

Zusammen mit der `isPlaying`-Variablen können Sie nun zu jeder Zeit den momentanen Status des Players abfragen.

Um die jeweiligen Schaltflächen zu »aktivieren«, erweitern Sie den Konstruktor in der ersten Zeile um einen Methodenaufruf zur Initialisierung der Steuerelemente:

```
initControls();
```

Um bei einem möglichen Stoppen des Abspielens die Anzeige des Players nicht in einem Zwischenzustand zu belassen, fügen Sie zwei neue Zeilen am Ende der `resetCurrentSound`-Methode ein.

Schritt 20

Klassen und Variablen für die Soundsteuerung



Zwischenergebnis:
30_AUDIO-PLAYER •
PLAYER_STEP20.AS

Schritt 21

Soundsteuerung umsetzen



Zwischenergebnis:
30_AUDIO-PLAYER •
PLAYER_STEP21.AS

```
positionBar_mc.width = 0;
updateTime(0);
```

Tipp

Wenn Sie die Standard-Einstellung im ACTION-SCRIPT 3.0-EINSTELLUNGEN-Fenster nicht verändert haben, sollte die Funktion BÜHNENINSTANZEN AUTOMATISCH DEKLARIEREN aktiv sein. Somit sind die Instanzen der Schaltflächen auf der Bühne innerhalb der Dokumentklasse ohne weiteres Zutun über ihren Instanznamen ansprechbar. Sie müssen diese mit Flash CS3 (im Gegensatz zu Flash 8) nicht noch als Klassenvariablen deklarieren.

Die beiden Zeilen sorgen dafür, dass beim Stoppen zum einen der Statusbalken und zum anderen die Zeit wieder auf Null gesetzt werden.

Nun aber zurück zur eigentlichen Umsetzung der Player-Steuerung. Die `initControls`-Methode, die Sie bereits aus dem Konstruktor heraus aufrufen, fehlt noch in der Player-Klasse. Wie immer hängen Sie diese Methode unten in der Klasse an:

```
1: private function initControls():void
2: {
3:     stop_btn.addEventListener(MouseEvent.CLICK, onClick);
4:     play_btn.addEventListener(MouseEvent.CLICK, onClick);
5:     pause_btn.addEventListener(MouseEvent.CLICK, onClick);
6:     next_btn.addEventListener(MouseEvent.CLICK, onClick);
7:     prev_btn.addEventListener(MouseEvent.CLICK, onClick);
8: }
```

Jedem auf der Bühne befindlichen Steuerelement wird in dieser Funktion ein `CLICK`-Ereignis-Listener zugewiesen (Zeilen 3 bis 7). Beim Klicken des Benutzers auf eine der Schaltflächen wird in allen Fällen die `onClick`-Methode aufgerufen. Diese übernimmt dann die weitere Verarbeitung des Ereignisses.

Es folgen die zur Steuerung benötigten Methoden:

```
9: private function onClick(evt:MouseEvent):void
10: {
11:     switch(evt.target)
12:     {
13:         case stop_btn: stopSound(); break;
14:         case play_btn:
15:             if(isPausing) pausePlay();
16:             else if(!isPlaying) playSound(currentIndex);
17:             else stopSound(), playSound(currentIndex);
18:             break;
19:         case pause_btn: if(isPlaying) pausePlay(); break;
20:         case next_btn: playNext(); break;
21:         case prev_btn: playPrev(); break;
22:     }
23: }
```


Wie bereits erklärt, fängt die `onClick`-Methode alle `CLICK`-Ereignisse der Player-Steuerung ab, überprüft, welches Element das Ereignis ausgelöst hat und reagiert dann mit den entsprechenden Methodenaufrufen.

Der Stopp-Button ruft in Zeile 13 die `stopSound`-Methode auf.

Beim Play-Button muss weiter unterschieden werden, ob der Song gerade pausiert (Zeile 15), noch nicht begonnen hat (Zeile 16) oder bereits abspielt (Zeile 17). Pausiert der Song, wird die noch fehlende Methode `pausePlay()` aufgerufen, die das Abspielen wieder anstößt. In dem Fall, dass der Song noch nicht begonnen hat, starten Sie die Musik mit dem Aufruf der `playSound`-Methode. Ein Sonderfall ist nun noch das Drücken während der Wiedergabe-Taste, die den Song wieder von Anfang an beginnen lässt: erst wird `stopSound()`, und gleich danach `playSound()` aufgerufen. In dieser Zeile müssen Sie darauf achten, dass ein Komma, und kein Semikolon, zwischen den beiden Anweisungen steht. Andernfalls gehört die `playSound`-Methode nicht mehr in den `else`-Teil der `if`-Anweisung und wird bei jedem Klick auf den `play_btn` aufgerufen, was wiederum zu Fehlern führen würde.

Das Ereignis des Pause-Buttons wird – nur wenn der Player gerade abspielt – über die `pausePlay`-Methode weiterverarbeitet.

Der Vorwärts-Button ruft in Zeile 20 die bereits bestehende Methode `playNext()` auf.

Zu guter Letzt sorgt der Aufruf der `playPrev`-Methode des Zurück-Buttons in Zeile 21 dafür, dass der vorherige Song abgespielt wird.

Nun folgen die noch nicht implementierten Steuerungs-Methoden:

```
24: private function pausePlay():void
25: {
26:     if(!isPausing)
27:     {
28:         currentChannel.stop();
29:     }
30:     else
31:     {
32:         currentChannel = currentSound.play(sPosition);
33:         currentChannel.addEventListener(Event.SOUND_COMPLETE,
34:             onSoundComplete);
35:     }
36:     isPausing = !isPausing;
37: }
38: private function playPrev():void
39: {
40:     stopSound();
```

```

40:     currentIndex = (currentIndex > 0) ? --currentIndex :
        songs.length-1;
41:     playSound(currentIndex);
42: }

```

In der `stopSound`-Methode müssen Sie zusätzlich zu der `isPlaying`-Variable auch die neue `isPausing`-Variable auf `false` setzen:

```
isPlaying = isPausing = false;
```

Die `pausePlay`-Methode sorgt dafür, dass der Song entweder anhält, wenn er vorher abgespielt wurde (ab Zeile 26), oder umgekehrt, dass er von der letzten Position wieder abspielt, wenn er vorher pausierte (ab Zeile 30). Beim erneuten Starten des Songs müssen Sie darauf achten, dass Sie anschließend in Zeile 33 den `SOUND_COMPLETE`-Listener neu zuweisen. Zudem wird in Zeile 35 nach jedem Aufruf der Methode die `isPausing`-Variable negiert, d. h. von `true` auf `false` gesetzt oder umgekehrt.

Die `playPrev`-Methode ab Zeile 37 ähnelt sehr der bereits bestehenden Methode `playNext()` – der einzige Unterschied ist, dass sie den Songindex nicht hoch-, sondern herunterzählt (Zeile 40). Sollte der Index kleiner als 0 sein, springt er auf den letzten Song in Ihrer Playliste: `songs.length-1`.

Speichern Sie die Änderungen in `PLAYER.AS` mit `Strg/⌘+S`.

Schritt 22

Flash-Film testen

Wenn Sie den Flash-Film nun über `Strg/⌘+↵` testen, können Sie das Abspielverhalten des Players über die Steuerelemente auf der Bühne beeinflussen. Spielen Sie ein bisschen mit den einzelnen Schaltflächen herum, so dass Sie die Auswirkungen aller Buttons kennenlernen.



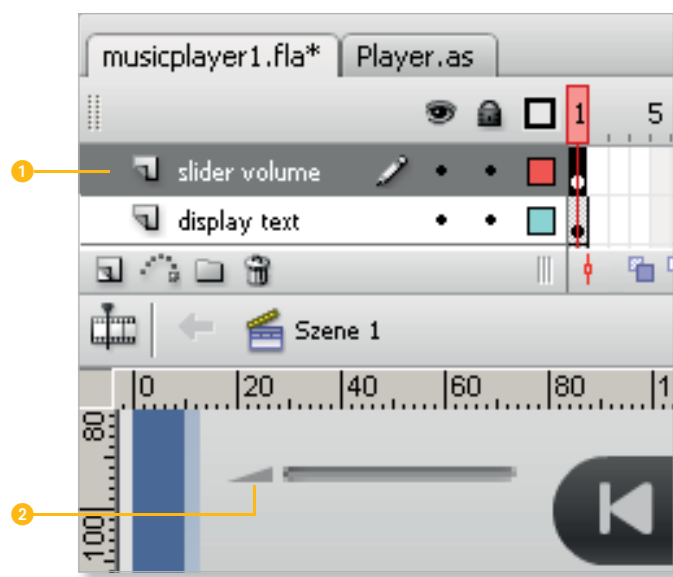
Abbildung 16 ►

Die Buttons funktionieren: Der MP3-Player hat die wichtigsten Funktionen bekommen.

Was bei keinem Audio-Player fehlen darf, ist der Lautstärkeregler. Ohne diesen hat der Benutzer keine Möglichkeit, seinen Lieblingssong lauter zu hören oder die »Schnulze« der Freundin leiser zu stellen.

Der Lautstärkeregler unseres Players wird per Drag&Drop bedienbar sein und sich in der Beispieldatei links neben den anderen Steuerelementen befinden.

Wählen Sie in Ihrer Flash-Datei die oberste Ebene aus, und fügen Sie über EINFÜGEN • ZEITLEISTE • EBENE eine weitere Ebene ein. Diese sollte automatisch oberhalb der ausgewählten Ebene erstellt worden sein. Doppelklicken Sie auf den Namen der Ebene, und benennen Sie sie mit »slider volume« ❶. Anschließend legen Sie die Hintergrundgrafik des Reglers an. Im Beispiel wird mit einer einfachen png-Datei gearbeitet. Importieren Sie die Datei SLIDER_BG.PNG aus dem ASSETS-Ordner auf der DVD über [Strg]/[⌘]+[R] auf die Bühne, und platzieren Sie sie auf der Position x = 29, y = 92. Damit der Benutzer weiß, worum es sich bei diesem Regler handelt, zeichnen Sie ein kleines Lautstärkesymbol ❷ links neben die soeben platzierte Grafik auf die gleiche Ebene.



Schritt 23

Lautstärke regulieren

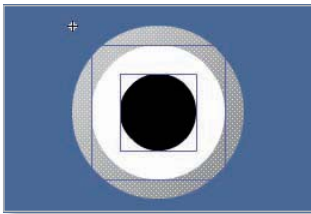
◀ **Abbildung 17**
Slider-Grafik für die
Lautstärkeregelung

Jetzt fehlt nur noch der Knopf des Reglers, der später vom Benutzer hin- und hergezogen werden kann. Hierfür zeichnen Sie einen grauen Kreis mit einem Durchmesser von 9 Pixel.

Konvertieren Sie den Kreis über [F8] zu einem Movieclip mit dem Namen »btn slider«. Über den bestehenden Kreis zeichnen Sie zwei

Schritt 24

Reglerknopf hinzufügen



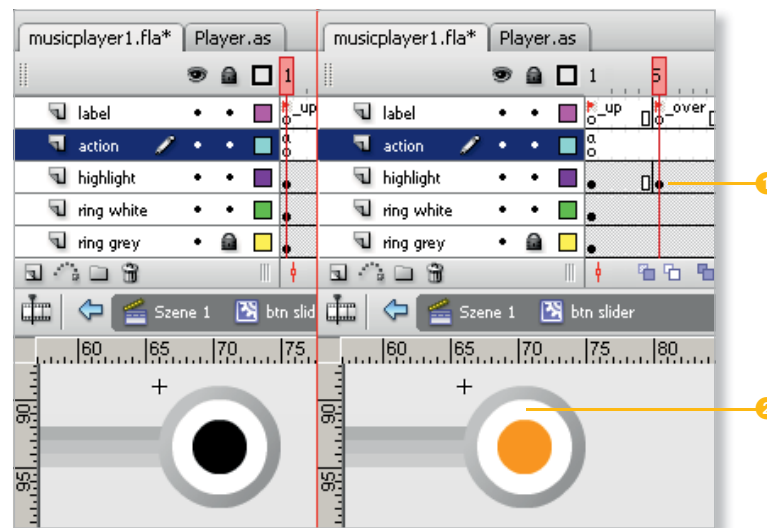
▲ **Abbildung 18**
Der fertige Button
(vergrößert)

weitere, wobei der zweite weiß und 7 Pixel groß sein sollte. Der letzte und kleinste Kreis sollte schwarz sein und einen Durchmesser von 4 Pixel haben.

Erstellen Sie innerhalb des Movieclips zwei weitere Ebenen »action« und »label«, und erweitern Sie die Zeitleiste aller Ebenen um 14 Bilder. Erzeugen Sie nun über [F6] in Bild 5 und 10 auf der »label«-Ebene zwei neue Schlüsselbilder. Geben Sie Bild 1 der »label«-Ebene den Namen »_up«; die beiden neu erstellten Schlüsselbilder erhalten die Namen »_over« und »_down«.

Schließlich erstellen Sie auf der Ebene, auf der der kleinste Kreis liegt, in Bild 5 ein weiteres Schlüsselbild ① und ändern dort die Farbe des Kreises auf Orange ②.

Abbildung 19 ▶
Für den Rollover-Effekt
setzen Sie den Knopf
auf Orange.



buttonMode

Ist der `buttonMode` eines Movieclips auf `true` gesetzt, verhält sich dieser wie ein Button und zeigt beim Darüberfahren mit der Maus einen Hand-Cursor an. Wenn der Movieclip die Sprungmarken »_up«, »_over« bzw. »_down« beinhaltet, wird zudem die jeweilige Sprungmarke angesprungen.

In das erste Bild der »action«-Ebene kommt der ActionScript-Befehl `stop()`; – öffnen Sie über [F9] bzw. [Alt]+[F9] das AKTIONEN-Fenster, und tragen Sie den Befehl dort in der ersten Zeile ein. Durch erneutes Drücken der Tastenkombination schließen Sie das Fenster wieder.

Wechseln Sie zurück auf die Bühne, und geben Sie dem eben erstellten Movieclip den Instanznamen »sliderVolume_mc«.

Movieclip statt Schaltfläche

Sicherlich fragen Sie sich, ob es hier nicht sinnvoll wäre, eine Schaltfläche anzulegen, und keinen Movieclip, da der Knopf sich genau wie ein Button verhalten und beim Darüberfahren mit der Maus den Zustand wechseln soll. In diesem Fall müssen Sie jedoch den Typ `MovieClip` wählen, da nur Movieclips über die benötigten `startDrag`- und `stopDrag`-Methoden verfügen.

Speichern Sie mit `Strg/Cmd + S` die eben ausgeführten Änderungen in der Datei `MUSICPLAYER1.FLA`, und wechseln Sie anschließend wieder in den Editor zu `PLAYER.AS`. Hier werden Sie nun die Lautstärkefunktion erstellen und mit dem Regler auf der Bühne verknüpfen. Um dies möglich zu machen, müssen Sie die folgenden beiden externen Klassen importieren:

```
import flash.geom.Rectangle;
import flash.media.SoundTransform;
```

Zudem benötigen Sie einige neue Variablen, die Sie unterhalb der bereits bestehenden Klassenvariablen anordnen:

```
private const SLIDER_VOL_X:int = 30;
private const SLIDER_VOL_Y:int = 89;
private const SLIDER_DRAG_WIDTH:int = 36;
```

Die drei Konstanten definieren den Bereich, in dem der Lautstärkeregler per Drag & Drop bewegt werden kann.

Jetzt fehlen Ihnen noch die Methoden, die dafür sorgen, dass der Benutzer den Lautstärkeregler bewegen kann. Fügen Sie als Erstes die Mouse-Listener ans Ende der `initControls`-Methode ein:

```
sliderVolume_mc.addEventListener(MouseEvent.CLICK, dragVolume);
sliderVolume_mc.addEventListener(MouseEvent.CLICK, dropVolume);
sliderVolume_mc.buttonMode = true;
stage.addEventListener(MouseEvent.CLICK, dropVolume);
```

Die neuen Listener in den ersten beiden Zeilen bewirken, dass beim Klicken bzw. beim Loslassen der Maustaste die `dragVolume`- bzw. `dropVolume`-Methode aufgerufen wird. In der nächsten Zeile aktivieren Sie den vorher besprochenen `buttonMode` des Reglers.

Die letzte hinzugefügte Zeile ist ein »Workaround«: Da es in ActionScript 3 keinen `onReleaseOutside`-Listener mehr gibt, fangen Sie das Loslassen der Maustaste außerhalb der Schaltfläche mit einem globalen Stage-Listener ab. Dieser reagiert immer, wenn die Maustaste nach dem Drücken wieder losgelassen wird, und delegiert den Aufruf an die Methode `dropVolume()`.

Schritt 25

Lautstärkefunktion umsetzen



Zwischenergebnis:
30_AUDIO-PLAYER •
PLAYER_STEP25.AS

Schritt 26

Drag & Drop implementieren



Zwischenergebnis:
30_Audio-Player •
Player_step26.as

Die folgenden Methoden fügen Sie unter die letzte Funktion der Player-Klasse ein:

```
1: private function dragVolume(evt:MouseEvent):void
2: {
3:     var tRect:Rectangle = new Rectangle(SLIDER_VOL_X,
        SLIDER_VOL_Y,SLIDER_DRAG_WIDTH,0);
4:     sliderVolume_mc.startDrag(false, tRect);
5:     stage.addEventListener(MouseEvent.MOUSE_MOVE,
        onDragVolume);
6: }
```

Die `dragVolume`-Methode wird bei einem Klick auf den Regler aufgerufen und startet in Zeile 4 das Drag & Drop. In der Zeile davor wird ein Rechteck festgelegt, in dem der Benutzer den Regler bewegen kann. Hier kommen die Konstanten aus dem vorherigen Schritt wieder ins Spiel – sie definieren den oberen linken und den unteren rechten Punkt des Rechtecks. Mit dem globalen `MOUSE_MOVE`-Listener in Zeile 5 wird bei jeder Bewegung des Lautstärkereglers die `onDragVolume`-Methode aufgerufen.

Fügen Sie direkt im Anschluss folgenden Code an:

```
7: private function dropVolume(evt:MouseEvent=null):void
8: {
9:     sliderVolume_mc.stopDrag();
10:    stage.removeEventListener(MouseEvent.MOUSE_MOVE,
        onDragVolume);
11: }
```

Weil die `dropVolume`-Methode nicht nur per Ereignis-Listener aufgerufen wird, müssen Sie in Zeile 7 beachten, dass der Übergabeparameter optional ist und ihn somit schon in der Methodendeklaration auf `null` setzen. In der Methode stoppen sie als Erstes in Zeile 9 das Drag & Drop des Reglers. Anschließend entfernen Sie den globalen `MOUSE_MOVE`-Listener wieder.

```
12: private function onDragVolume(evt:MouseEvent):void
13: {
14:     setVolume((sliderVolume_mc.x - SLIDER_VOL_X) /
        SLIDER_DRAG_WIDTH);
15: }
```

Die `onDragVolume`-Methode ab Zeile 12 wird immer dann aufgerufen, wenn der Regler durch den Benutzer bewegt wird. Sie berechnet in Zeile 14 die prozentuale Position des Reglers – einen Wert zwischen 0 und 1 – und reicht diese an die `setVolume`-Methode weiter.

```
16: private function setVolume(vol:Number):void
17: {
18:     currentVolume = vol;
19:     currentChannel.soundTransform = new
        SoundTransform(currentVolume);
20: }
```

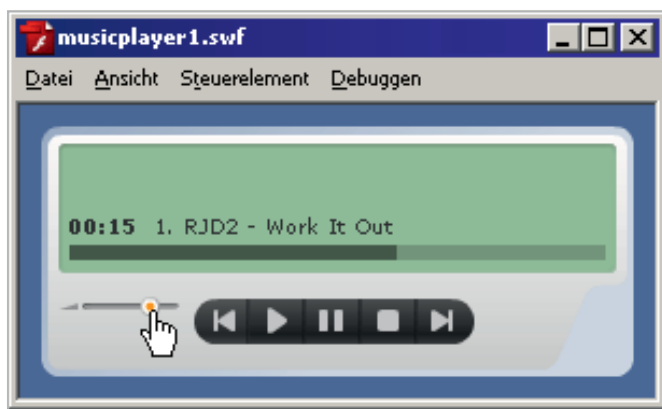
Die `setVolume`-Methode ab Zeile 16 erwartet eine Zahl zwischen 0 und 1 als Übergabeparameter, der in Zeile 18 in der `currentVolume`-Variable gespeichert wird. Danach benutzen Sie in Zeile 19 ein neues `SoundTransform`-Objekt dazu, die Lautstärke des laufenden Songs zu ändern.

Da Sie innerhalb des Skripts die Lautstärke nur während des Abspielens ändern können, müssen Sie darauf achten, dass Sie bei jedem neuen Starten eines Songs die Lautstärke erneut regulieren. Fügen Sie zu diesem Zweck folgende Zeile ans Ende der `playSound`-Methode ein:

```
setVolume(currentVolume);
```

Speichern Sie die Änderungen der Player-Klasse mit `(Strg)/(⌘)+[S]` ab.

Um Ihren Lautstärkeregler in Aktion zu sehen, testen Sie über `(Strg)/(⌘)+[↩]` den Flash-Film. Versuchen Sie nun, den Regler mit der Maus nach rechts und links zu ziehen, und achten Sie dabei auf die Lautstärke des abgespielten Songs.



Schritt 27

Testen des Flash-Films

◀ Abbildung 20

Der Mauszeiger wird zu einer kleinen Hand, und die Lautstärke kann justiert werden.

Schritt 28

Lautstärke visualisieren

Die SoundChannel-Klasse bietet Ihnen zudem die Möglichkeit, die Lautstärke des momentan abgespielten Sounds zu überwachen. Hierbei wird die Lautstärke in den linken und rechten Kanal aufgeteilt, da beide Kanäle unterschiedlich laut wiedergegeben werden können.

In diesem Schritt stellen Sie die beiden Kanäle des SoundChannel-Objekts grafisch in der linken oberen Ecke der Anzeige des Players dar. Zu diesem Zweck legen Sie drei neue Ebenen in MUSICPLAYER1.FLA an und benennen sie mit: »channel bg«, »channels« und »channel mask«.



Abbildung 21 ►
So sieht die Zeitleiste des Workshops bisher aus.

Auf der »channel bg«-Ebene zeichnen Sie nun zehn 15x1 Pixel große Rechtecke in der Farbe #7B997B, die mit einem Abstand von je 1 Pixel übereinanderliegen. Kopieren Sie anschließend alle zehn Balken, und fügen Sie die Kopien jeweils rechts neben den ersten Balken ein – auch wieder mit einem Abstand von 1 Pixel.

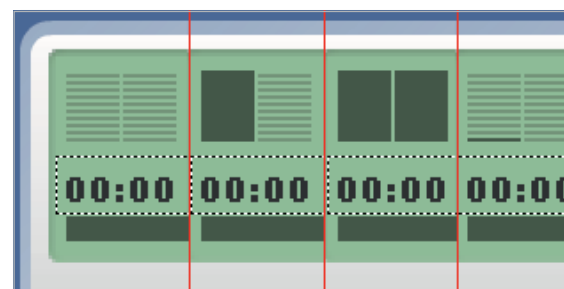
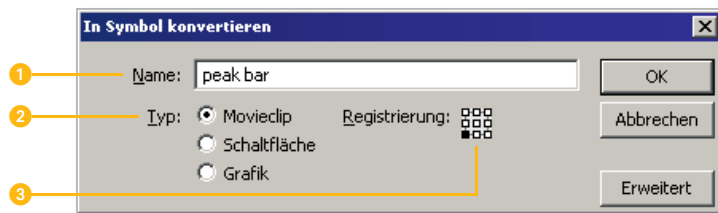


Abbildung 22 ►
Eine grafische Anzeige für die Soundlautstärke entsteht.

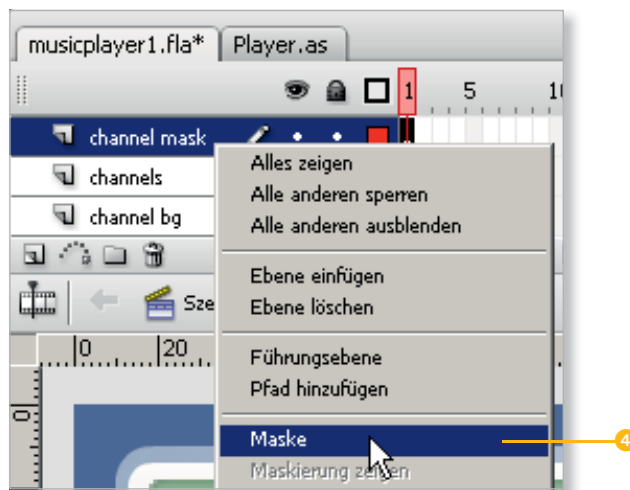
Auf der darüberliegenden Ebene »channels« zeichnen Sie ein 15x20 Pixel großes Rechteck in der Farbe #4A5C4A, das die zehn darunterliegenden

Balken komplett verdeckt. Konvertieren Sie das Rechteck nun über **[F8]** zu einem **MOVIECLIP** ②. Hierbei ist es wichtig, darauf zu achten, dass die **REGISTRIERUNG** ③ nicht wie sonst in der oberen linken, sondern in der unteren linken Ecke liegt. Geben Sie dem Symbol den **NAMEN »peak bar«** ①, und bestätigen Sie alles mit **OK**.



◀ **Abbildung 23**
In Symbol konvertieren

Jetzt kopieren Sie den eben erstellten Movieclip und legen ihn über die noch freien Balken. Setzen Sie den Instanznamen des linken Rechtecks auf »lPeakBar_mc« und den des rechten auf »rPeakBar_mc«. Als Letztes kopieren Sie nun die 20 Balken der »channel bg«-Ebene auf die »channel mask«-Ebene und setzen über das Kontextmenü (per Rechtsklick) diese Ebene auf **MASKE** ④.



◀ **Abbildung 24**
Die »channels«-Ebene wird durch die »channel mask«-Ebene maskiert.

Damit die beiden im vorherigen Schritt angelegten Balken auch wirklich den Ausschlag der Lautstärke des jeweiligen Kanals widerspiegeln, erweitern Sie die `onEnterFrame`-Methode der `Player`-Klasse unterhalb der ersten `if`-Anweisung um die folgenden fünf Zeilen:

```
1:  if(!isPausing)
2:  {
```

Schritt 29

Amplitude der Lautstärke berechnen

Zwischenergebnis:
30_AUDIO-PLAYER •
PLAYER_STEP29.AS



```
3:     lPeakBar_mc.height = Math.round(20 * currentChannel.  
    leftPeak);  
4:     rPeakBar_mc.height = Math.round(20 * currentChannel.  
    rightPeak);  
5: }
```

Da die `onEnterFrame`-Methode während des Abspielens 25 Mal pro Sekunde aufgerufen wird, ändern Sie mit diesen Zeilen auch genauso oft die Höhe – in diesem Fall den Ausschlag – der beiden Lautstärkebalken. Dabei wird die jeweilige Höhe in den Zeilen 3 und 4 über die `leftPeak`- bzw. `rightPeak`-Eigenschaft des `SoundChannel`-Objekts berechnet. Wenn Sie diesen Wert, der zwischen 0 und 1 liegt, mit der ursprünglichen und maximalen Höhe der Lautstärkebalken von 20 Pixeln multiplizieren und daraufhin zu einer Ganzzahl runden, erhalten Sie eine Höhe von 0 bis 20 Pixel.

Schritt 30 Flash-Film testen

Speichern Sie die Änderungen in `PLAYER.AS` mit `[Strg]/[⌘]+[S]`, und testen Sie den erweiterten Player mit `[Strg]/[⌘]+[↵]`. Sie sollten den Lautstärkeauschlag sofort erkennen.

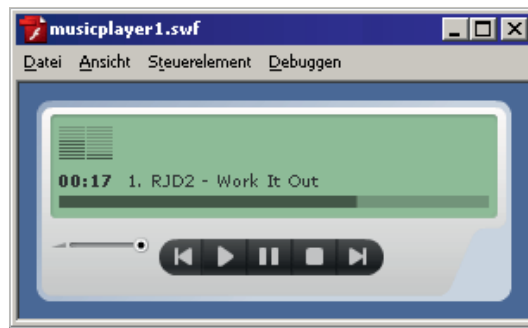
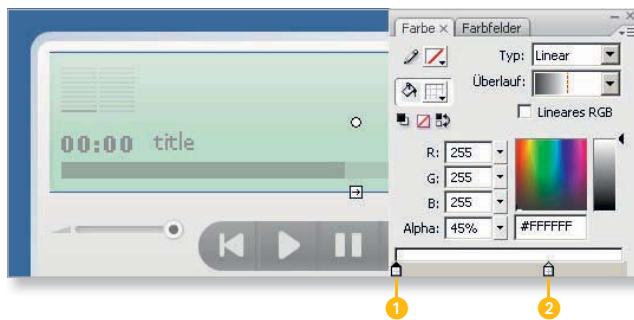


Abbildung 25 ►
Der linke und rechte Kanal
schlagen aus.

Schritt 31 Der letzte Schliff

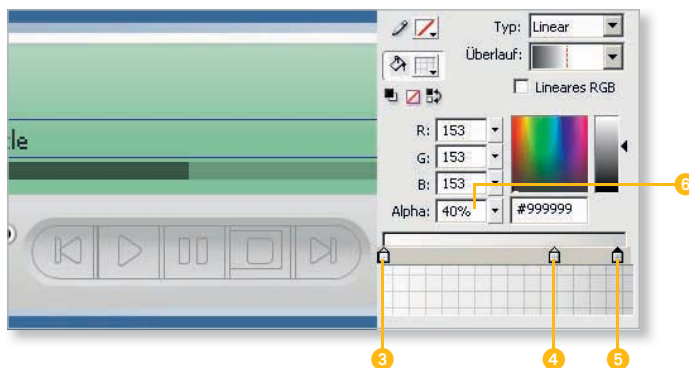
Um dem Player ein noch besseres Aussehen zu verpassen, wechseln Sie ein letztes Mal zu Ihrer Flash-Datei `MUSICPLAYER1.FLA`. Sie werden nun die Anzeige des Players mit einem leichten »Glow« aufwerten und zudem die Steuerelemente besser vom Hintergrund abheben.

Für den »Glow« erzeugen Sie eine weitere Ebene, »display glow«, die über allen bisherigen Ebenen liegt. Hier zeichnen Sie ein 259 x 60 Pixel großes Rechteck, das Sie genau über der Anzeige des Players platzieren. Als Farbe wählen Sie einen linearen Verlauf von 45% Weiß oben ① nach 0% Weiß unten ②.



◀ **Abbildung 26**
Ein linearer Verlauf bildet die Basis für ein schickeres Aussehen des MP3-Players.

Die Steuerelemente unterlegen Sie mit einem Rechteck mit stark abgerundeten Ecken, so dass der Eindruck entsteht, die Buttons seien in die Oberfläche des Players eingelassen. Bevor Sie damit anfangen, erzeugen Sie eine neue Ebene unterhalb der Steuerelemente. Das Rechteck sollte 138 x 27 Pixel groß sein und als Farbe einen linearen Verlauf haben – diesmal aber von Dunkelgrau (#999999) 5 über Hellgrau (#cccccc) 4 nach Weiß 3. Setzen Sie ALPHA bei allen drei Farben auf nicht mehr als 40 % 6.



◀ **Abbildung 27**
Die Buttons bekommen einen Rahmen.

Speichern Sie die fla-Datei ein letztes Mal mit `Strg/⌘+S` ab, und testen Sie die swf-Datei mit `Strg/⌘+↵`. Fertig ist Ihr voll funktionsfähiger MP3-Player!



◀ **Abbildung 28**
Der Player ist fertig.

Schritt 32

Flash-Film testen

Noch mehr Features?

Wenn Sie den Player um zusätzliche Funktionen erweitern möchten, schauen Sie sich die folgenden Workshops an. Dort finden Sie noch mehr interessante neue Features.

Soundspektrum

Audio-Player um eine Spektrumsanzeige erweitern

```
package
{
    import flash.display.MovieClip;
    import flash.events.Event;
    import flash.utils.ByteArray;
    import flash.media.SoundMixer;
    public class Spectrum extends MovieClip
    {
        private var bytes:ByteArray = new ByteArray();
        private var sBarArray:Array = new Array();
        public function Spectrum()
        {
            for(var i:int = 0; i < 64; i++)
            {
                var s:Spectrumstripe = new Spectrumstripe();
                s.x = i*3;
                addChild(s);
                sBarArray.push(s);
            }
        }
    }
}
```

In Flash CS3 und AS3 ist es nun endlich möglich, sich das Spektrum des abgespielten Sounds direkt ausgeben zu lassen. In diesem Workshop lernen Sie, wie ein ByteArray ausgelesen und dargestellt wird.

Zielsetzungen:

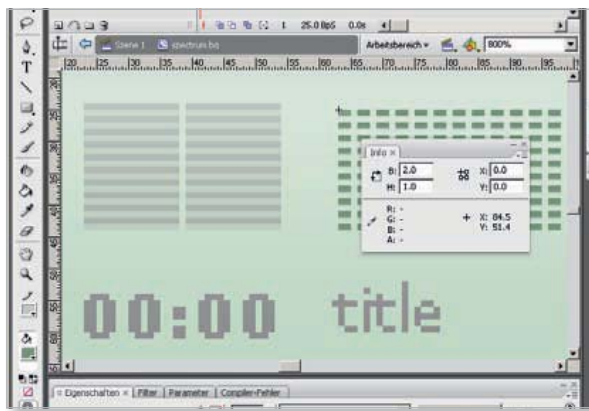
- Spektrumshintergrund zeichnen
- SoundMixer-Objekt anwenden
- Spektrum mit computeSpectrum auslesen
- Spektrum animieren

[Ergebnis: 31_SOUNDSPEKTRUM • MUSICPLAYER2.SWF]



Öffnen Sie die fla-Datei MUSICPLAYER.FLA aus dem vorherigen Workshop »Audio-Player«. Speichern Sie diese Datei unter dem Namen MUSICPLAYER2.FLA in einem neuen Ordner MUSICPLAYER2. Kopieren Sie auch die Datei PLAYER.AS aus dem vorigen Workshop in diesen Ordner.

Erstellen Sie mit EINFÜGEN • NEUES SYMBOL... ein neues Symbol mit dem Namen »spectrum bg«. In diesem positionieren Sie 2x1 Pixel große Quadrate mit der Farbe 0x7C987B. Es soll eine Matrix mit 64 Quadraten in der Horizontalen und 10 Quadraten in der Vertikalen entstehen. Der Abstand zwischen den Quadraten soll jeweils 1 Pixel betragen.



Schritt 1

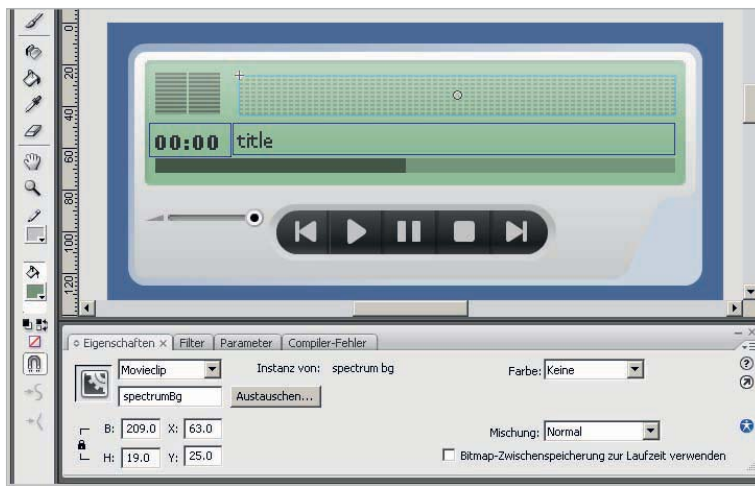
Flash-Datei abspeichern

Schritt 2

Spektrumshintergrund erstellen

◀ **Abbildung 1**
Der Hintergrund für die Spektrumsanzeige entsteht.

Ziehen Sie diesen Movieclip nun aus der Bibliothek auf die Ebene »channels«, und positionieren Sie ihn auf der X-Position 63 Pixel und der Y-Position 25 Pixel. Anschließend geben Sie dem Movieclip im Eigenschaftsfenster den Instanznamen »spectrumBg«.



◀ **Abbildung 2**
Der Movieclip wird aus der Bibliothek auf die Bühne gezogen und im MP3-Player platziert.

Schritt 3 Spektrumsmaske erstellen

Nun kopieren Sie den eben erstellten Movieclip mit **BEARBEITEN • KOPIEREN** oder **[Strg]/[⌘] + [C]** und fügen ihn mit **BEARBEITEN • AN POSITION EINFÜGEN** oder **[Strg]/[⌘] + [⇧] + [V]** an derselben Position ein. Geben Sie diesem neuen Movieclip den Instanznamen »spectrumMask«.

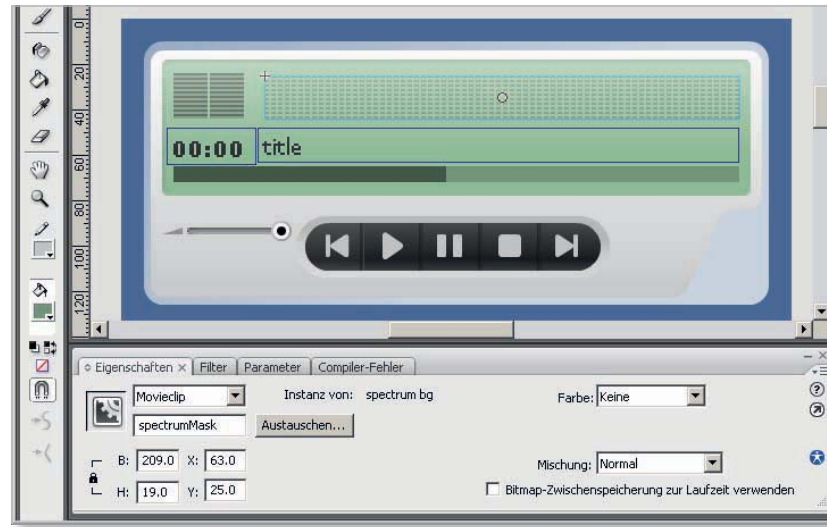


Abbildung 3 ►

Der Movieclip wird kopiert und an derselben Position ein zweites Mal eingefügt.

Schritt 4 Dokumentklasse erweitern

Zwischenergebnis:
31_SOUNDSPKTRUM •
PLAYER_STEP04.AS



Nun fügen Sie den Spectrum-Movieclip der Hauptzeitleiste hinzu. Deklarieren Sie dafür zunächst die Variable `spectrum_mc` in Zeile 34 der Datei `PLAYER.AS`:

```
34: private var spectrum_mc:Spectrum;
```

Fügen Sie ab Zeile 124 folgende Zeilen hinzu:

```
124: spectrum_mc = new Spectrum();  
125: spectrum_mc.x = 63;  
126: spectrum_mc.y = 44;  
127: addChildAt(spectrum_mc, getChildIndex(getChildByName(  
    "spectrumBg"))+1);  
128: spectrum_mc.mask = spectrumMask;
```

In den Zeilen 125 und 126 positionieren Sie den Movieclip an der korrekten Position.

Damit sich `spectrum_mc` über `spectrumBg` befindet (und nicht davon verdeckt wird), fügen Sie den Movieclip mit `addChildAt()` auf einer Tiefe ein, die höher als der Spectrum-Hintergrund liegt.

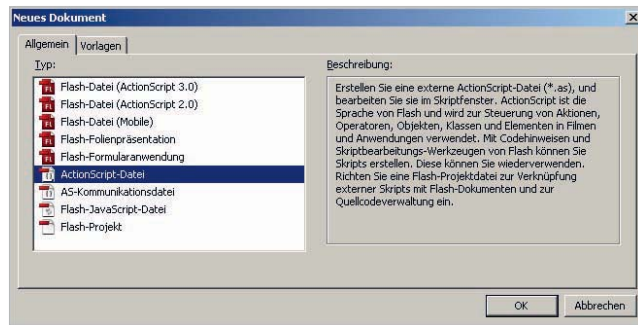
In Zeile 128 schließlich sorgen Sie dafür, dass der erstellte Movieclip `spectrum_mc` von dem Movieclip `spectrumMask` maskiert wird.

Nun muss noch ein Befehl eingefügt werden, der das Spektrum in jedem Frame aktualisiert. Ergänzen Sie Ihr Skript in Zeile 86 wie folgt:

```
86: spectrum_mc.drawSpectrum();
```

Wenn ein Sound abgespielt wird, wird nun in jedem Frame die (noch zu erstellende) Methode `drawSpectrum()` aufgerufen.

Erstellen Sie über **DATEI • NEU • ACTIONSCRIPT-DATEI** eine neue Datei, und speichern Sie diese unter dem Namen »Spectrum.as« im Ordner **MUSIC-PLAYER2**.



Schritt 5

Spectrum-Klasse erstellen

◀ **Abbildung 4**
Die Spectrum-Klasse wird angelegt.

Fügen Sie der Datei folgende Zeilen hinzu:

```
1: package
2: {
3:     import flash.display.MovieClip;
4:     import flash.events.Event;
5:     import flash.utils.ByteArray;
6:     import flash.media.SoundMixer;
7:     public class Spectrum extends MovieClip
8:     {
9:         private var bytes:ByteArray = new ByteArray();
10:        private var sBarArray:Array = new Array();
11:        public function Spectrum()
12:        {
13:        }
14:        public function drawSpectrum():void
15:        {
```

Schritt 6

Grundgerüst



Zwischenergebnis:
31_SOUNDSPEKTRUM •
SPECTRUM_STEP06.AS

```

16:         }
17:     }
18: }

```

In den Zeilen 3 bis 6 importieren Sie die später benötigten Klassen. In den Zeilen 9 und 10 deklarieren Sie die benötigten Variablen `bytes` und `sBarArray`.

Schritt 7 Spektrumstreifen anfügen

Zwischenergebnis:
31_SOUNDSPEKTRUM •
SPECTRUM_STEP07.AS



Fügen Sie im Konstruktor folgende Codezeilen hinzu:

```

1:  for(var i:int = 0; i < 64; i++)
2:  {
3:      var s:Spectrumstripe = new Spectrumstripe();
4:      s.x = i*3;
5:      addChild(s);
6:      sBarArray.push(s);
7:  }

```

Da das Spektrum 64 Streifen enthalten soll, wird in Zeile 1 eine `for`-Schleife 64-mal durchlaufen.

In Zeile 3 erstellen Sie in jedem Durchlauf der `for`-Schleife eine neue Instanz eines Spektrumstreifens. Diese positionieren Sie in Zeile 4 immer 3 Pixel entfernt von dem vorherigen Streifen. Um diesen Movieclip anzuzeigen, fügen Sie ihn mit `addChild()` der Zeitleiste hinzu.

In Zeile 6 schließlich speichern Sie eine Referenz des Movieclips im Array `sBarArray`, um später auf den Movieclip zugreifen zu können.

Schritt 8 Spektrum-Bytes auslesen

Zwischenergebnis:
31_SOUNDSPEKTRUM •
SPECTRUM_STEP08.AS



Flash stellt ab Flash CS3 eine Funktion bereit, die den Soundstream auslesen kann. Ergänzen Sie Ihr Skript in der Funktion `drawSpectrum()` um folgenden Code:

```
SoundMixer.computeSpectrum( bytes, true, 1 );
```

In dieser Zeile wird das `ByteArray` `bytes` mit den Byte-Informationen des aktuellen Sounds befüllt.

Schritt 9 Links-/Rechts-Arrays erstellen

Da wir das Spektrum für den rechten und linken Kanal getrennt auslesen werden, benötigen wir pro Kanal ein Array:


```
var floatLeftArray:Array = new Array();
var floatRightArray:Array = new Array();
```

Ergänzen Sie Ihr Skript unter der in Schritt 8 eingefügten Codezeile wie folgt:

```
1: for(var i:int = 0; i < 512; i++)
2: {
3:     if(i < 256) floatLeftArray.push(bytes.readFloat());
4:     else floatRightArray.push(bytes.readFloat());
5: }
```

Da das ByteArray 512 Einträge hat, wird in Zeile 1 die `for`-Schleife 512-mal durchlaufen.

Die ersten 256 Bytes bilden den linken Kanal ab, die letzten 256 Bytes den rechten Kanal. In den Zeilen 3 und 4 werden die Einträge auf die entsprechenden Arrays verteilt. `readFloat()` gibt eine Zahl zwischen 0 und 1 zurück.

Nun werden Sie die Spektrumstreifen entsprechend dem ausgelesenen Spektrum skalieren. Ergänzen Sie eine zweite `for`-Schleife unter der in Schritt 10 eingefügten:

```
1: for(var k:int = 0; k < 64; k++)
2: {
3:     var val:Number = 0;
4:     for(var j:int = (k*4); j < ((k+1)*4); j++)
5:     {
6:         val += floatLeftArray[j] + floatRightArray[j];
7:     }
8:     sBarArray[k].scaleY = val/8;
9: }
```

Zuvor haben Sie 64 Streifen angefügt – daher wird in Zeile 1 die `for`-Schleife 64-mal durchlaufen.

In Zeile 3 wird eine lokale Variable zur Berechnung eines Durchschnittswerts initialisiert.



Zwischenergebnis:
31_SOUNDSPEKTRUM •
SPECTRUM_STEP09.AS

Schritt 10

ByteArray auslesen



Zwischenergebnis:
31_SOUNDSPEKTRUM •
SPECTRUM_STEP10.AS

Schritt 11

Spektrum animieren



Zwischenergebnis:
31_SOUNDSPEKTRUM •
SPECTRUM_STEP11.AS

256 geteilt durch 64 ergibt 4. Darum muss jeweils in Vierer-Schritten ein Wert errechnet werden. In Zeile 4 werden diese Vierer-Schritte in der `for`-Schleife angesprochen.

In Zeile 6 werden die Werte in die lokale Variable `val` addiert.

Schließlich wird in Zeile 8 der aktuelle Streifen mit dem Wert von `val` geteilt durch 8 skaliert.



Abbildung 5 ►

Testen Sie die Datei. Die Balken werden jetzt in Abhängigkeit vom Spektrum des Sounds skaliert.

Adobe AIR

Adobe AIR (Adobe Integrated Runtime) ist ein weiterer Baustein in der Rich Internet Application-Offensive von Adobe.

Es handelt sich dabei um eine betriebssystemübergreifende Laufzeitumgebung, die es Entwicklern ermöglicht, mit HTML/CSS, Ajax (JavaScript) oder Flash- bzw. Flex-Anwendungen zu erstellen. Diese sind auf dem Desktop des Nutzers betriebssystemunabhängig lauffähig.

Um AIR-Anwendungen einsetzen zu können, muss sich der Anwender ein spezielles Plug-in herunterladen, das kostenlos auf den Webseiten von Adobe verfügbar ist.

Wenn Sie AIR-Programme entwickeln möchten, haben Sie vielfältige Möglichkeiten. AIR ist Bestandteil von Flex Builder 2. Um aus einer Flex-Anwendung eine AIR-Anwendung zu machen, müssen Sie nur wenige Einstellungen verändern. Auch für Flash CS3 und Dreamweaver ist eine AIR-Erweiterung verfügbar.



◀ **Abbildung 1**
Das Adobe-AIR-Logo

Funktionen

AIR-Anwendungen laufen auf dem Desktop. Für die Nutzung wird, im Gegensatz zu herkömmlichen Flash-/Flex-Anwendungen, keine Internetverbindung benötigt. Wenn Sie also auf eine AIR-Anwendung doppelklicken, öffnet sich nicht der Webbrowser, sondern ein Programmfenster, wie Sie es von Ihrem Betriebssystem (Windows oder Mac) kennen.

Als Entwickler können Sie bestimmen, wie dieses Fenster (engl.: chrome) aussieht. So kann das Fenster z. B. transparent sein. Es wird dann nur Ihre Anwendung angezeigt, um alle üblichen Standardelemente (z. B. für das Schließen und Minimieren des Fensters) müssen Sie sich selbst kümmern.

Die Laufzeitumgebung (also das Plug-in) von AIR stellt Ihnen viele Funktionen zur Verfügung, die ein »richtiges« Desktop-Programm auch hat. So können Sie Ihr Programmfenster minimieren, schließen, weitere Fenster oder auch ein anderes auf dem Rechner installiertes Programm öffnen.

Außerdem haben Sie Zugriff auf das Dateisystem. Sie können Dateien und Ordner erstellen, verschieben, umbenennen oder löschen. Dies ermöglicht es Ihnen, in unserem Audio-Player-Beispiel (siehe Workshop »Audio-Player«) direkt auf die Dateien auf der Festplatte zuzugreifen und sie abzuspielen.

Erstellen

Sie können AIR-Anwendungen mit Flex, Flash oder Dreamweaver erstellen. Es ist allerdings auch möglich, AIR-Anwendungen ganz ohne Adobe-Programme zu entwickeln.

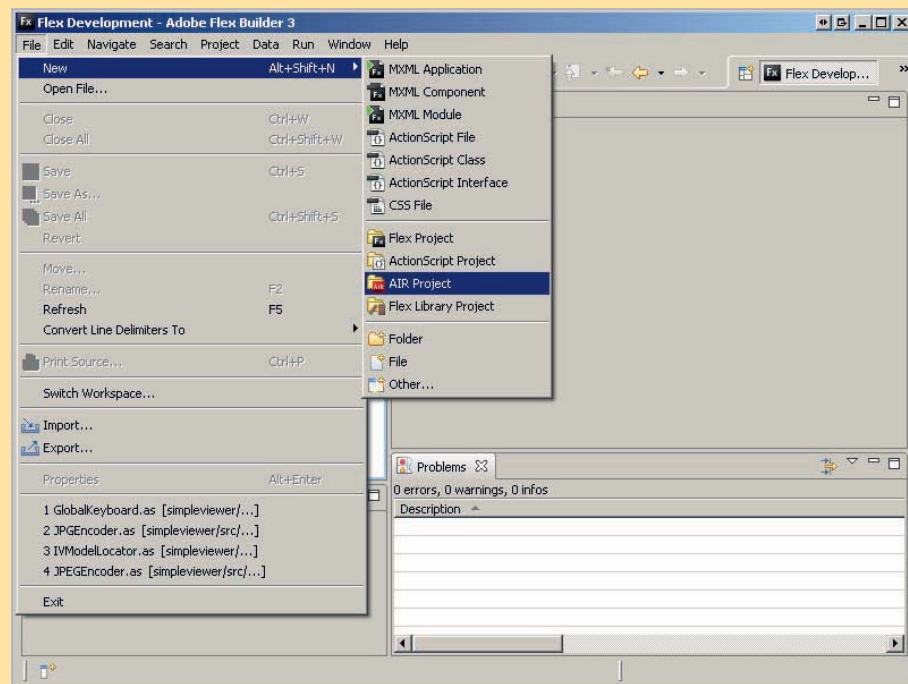


Abbildung 2 ►

AIR-Projekt in Flex erstellen

Dafür müssen Sie sich auf der Adobe-Website das Adobe AIR SDK herunterladen. Mit dem darin enthaltenen Kommandozeilen-Tool können Sie völlig kostenlos aus Ihren HTML- oder MXML-Dateien AIR-Anwendungen erstellen.

Einfacher geht es allerdings mit den Adobe-Programmen.

Veröffentlichen

AIR-Anwendungen benötigen ein Zertifikat. Dies stellt sicher, dass die Anwendung wirklich von Ihnen entwickelt wurde. Sie können sich selbst ein Zertifikat erstellen oder sich ein kommerzielles Zertifikat kaufen. Genaues erfahren Sie auf der Website von Adobe.

Anschließend können Sie die AIR-Datei auf Ihrer Website veröffentlichen oder Ihren Freunden per E-Mail zuschicken.

Fazit

Adobe AIR erlaubt es Webentwicklern, Offline-Anwendungen – »richtige« Programme – zu programmieren. Damit öffnet Adobe eine Tür, die sich sicherlich so schnell nicht wieder schließen wird.

Auch wenn C- oder Java-Programmierer viele Funktionen vermissen werden und die Geschwindigkeit noch etwas zu wünschen übrig lässt, so wird doch einer sehr großen Entwicklergemeinde die Möglichkeit gegeben, einfache Programme zu schreiben oder die Online-Anwendung auf den Desktop zu portieren.

Es macht Spaß, mit AIR herumzuspielen. Mal sehen, was Sie und uns da noch alles erwartet.

AIR-Musikplayer

Aus dem MP3-Player wird ein vollwertiges Offline-Programm

```
<?xml version="1.0" encoding="utf-8" ?>
<application appId="com.air.flash.musicplayer3" version="1.0"
xmlns="http://ns.adobe.com/air/application/1.0.M5">
  <name>airmusicplayer</name>
  <description></description>
  <title>airmusicplayer</title>
  <copyright>Unknown</copyright>
  <initialWindow>
    <title>airmusicplayer</title>
    <content>musicplayer3.swf</content>
    <systemChrome>none</systemChrome>
    <transparent>true</transparent>
    <visible>true</visible>
  </initialWindow>
  <!--<installFolder></installFolder>-->
  <!--<programMenuFolder></programMenuFolder>-->
  <icon>
  </icon>
  <!--<handleUpdates/>-->
</application>
```

Völlig neue Möglichkeiten eröffnet Adobe seiner Entwicklergemeinde mit dem Release von AIR. Ab jetzt kann jeder Programmierer, der sich auf HTML, JavaScript oder Flash versteht, Desktop-Anwendungen erstellen. In diesem Workshop lernen Sie, einen Audio-Player um Desktop-Funktionen zu erweitern.

Zielsetzungen:

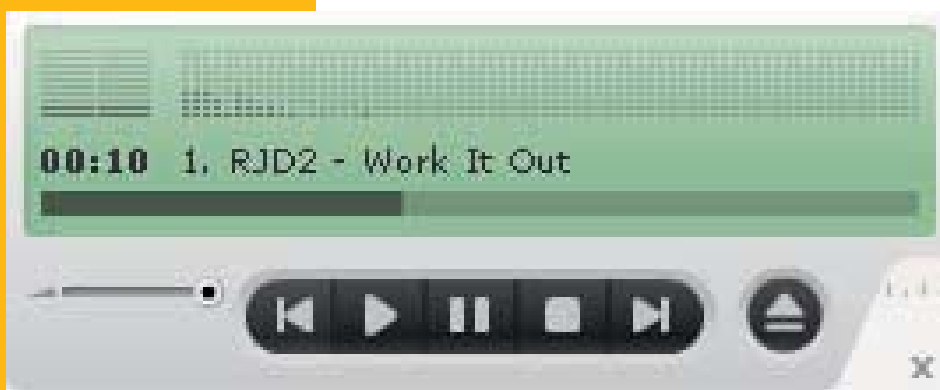
AIR-Parameter einstellen

Möglichkeit, die Anwendung zu schließen, integrieren

Möglichkeit, weitere Musikstücke hinzuzufügen, integrieren

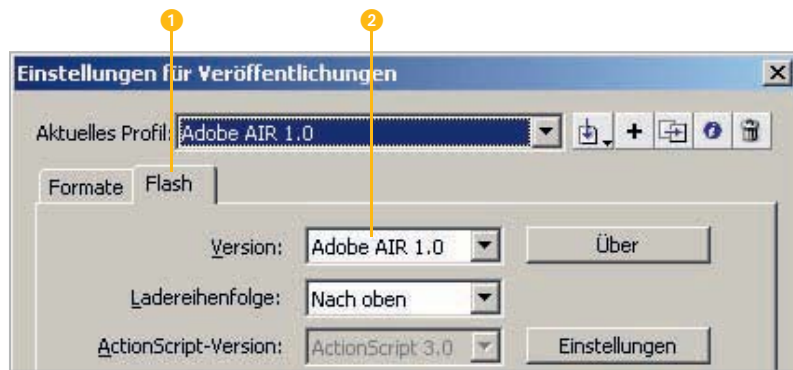
Drag-&-Drop-Funktionalität einbauen

[Ergebnis: 32_AIR-Musik-PLAYER • MUSICPLAYER3.SWF]



Erstellen Sie einen neuen Ordner mit dem Namen MUSICPLAYER3. Kopieren Sie alle Dateien aus dem vorherigen Workshop in diesen neuen Ordner, und öffnen Sie die Datei MUSICPLAYER2.FLA. Speichern Sie diese Datei über DATEI • SPEICHERN UNTER... unter dem Namen MUSICPLAYER3.FLA.

Öffnen Sie mit DATEI • EINSTELLUNGEN FÜR VERÖFFENTLICHUNGEN oder [Strg]/[Alt]+[⇧]+[F12] den Dialog zum Ändern der Exporteinstellungen. Im Reiter FLASH ❶ wählen Sie in der Liste VERSION ❷ den Eintrag ADOBE AIR 1.0. Bestätigen Sie mit OK.



Schritt 1

Dokument erstellen

Schritt 2

Exporteinstellungen setzen

◀ Abbildung 1

Als Exporteinstellung muss ADOBE AIR 1.0 gewählt sein, wenn Sie eine AIR-Anwendung erstellen möchten.

Öffnen Sie über BEFEHLE • AIR – APPLICATION AND PACKAGE SETTINGS die Einstellungen von Adobe AIR. In den Feldern FILE NAME ❸ und NAME ❹ geben Sie jeweils »airmusicplayer« ein (siehe Abbildung 2). Dies bewirkt, dass die exportierte AIR-Datei »airmusicplayer.air« heißt und im Installationsdialog die Anwendung mit »airmusicplayer« benannt wird.

Außerdem wählen Sie in der Liste WINDOW STYLE ❺ die Einstellung CUSTOM CHROME (TRANSPARENT). Die exportierte AIR-Anwendung ist nun transparent und hat nicht die üblichen Fensterleisten des Betriebssystems.

Zusätzlich müssen Sie für Ihre Anwendung noch die digitale Signatur angeben (mehr dazu im AIR-Exkurs auf Seite 377). Klicken Sie dafür auf den Button CHANGE... ❻ neben dem Feld DIGITAL SIGNATURE. In dem sich nun öffnenden Dialog wählen Sie PREPARE AN AIR INTERMEDIATE (AIRI) FILE THAT WILL BE SIGNED LATER ❼. Damit geben Sie an, dass Sie sich um die Signatur später kümmern werden. Dies bewirkt, dass Sie die erstellte AIR-Datei auf Ihrem Rechner testen können – sie wird aber nicht auf anderen Rechnern laufen.

Schritt 3

AIR-Einstellungen festlegen

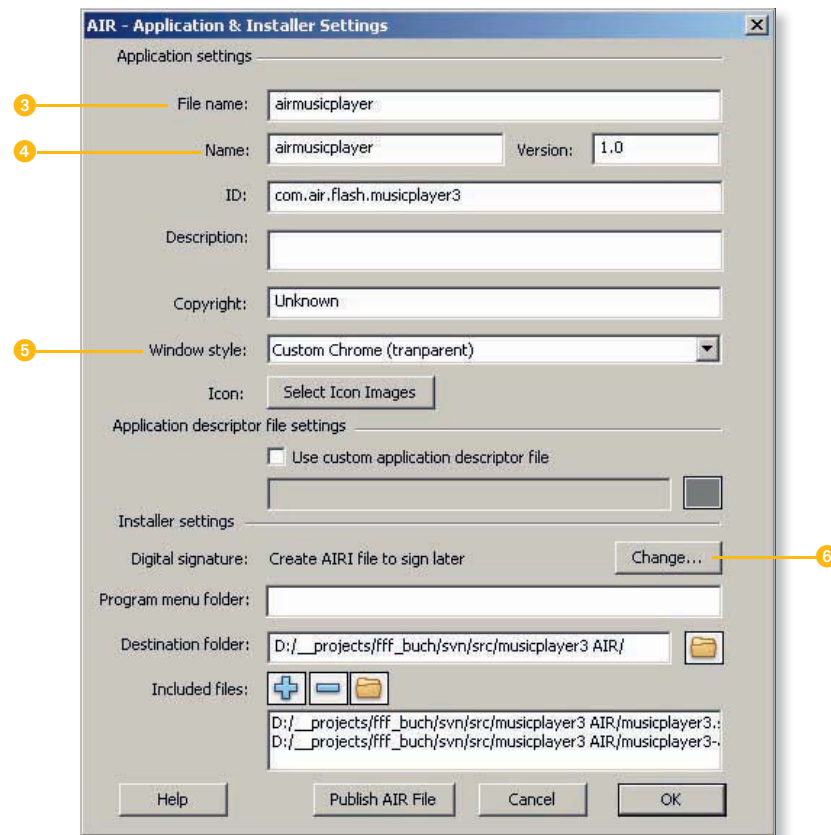


Abbildung 2 ►
In den AIR-Einstellungen legen Sie Namen und Aussehen Ihrer AIR-Anwendung fest.

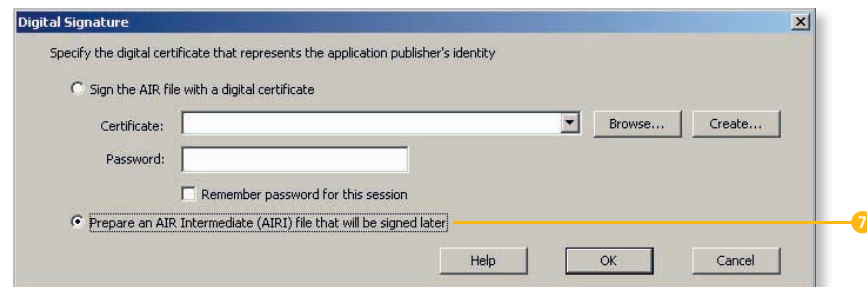


Abbildung 3 ►
Einstellungen für die digitale Signatur

Bestätigen Sie Ihre Angaben jeweils mit OK.

Schritt 4 Buttons hinzufügen

Öffnen Sie die Datei AIR_BUTTONS.FLA von der Buch-DVD. Kopieren Sie die beiden darin befindlichen Buttons auf die Ebene »btn« in Ihrer Datei MUSICPLAYER3.FLA.

Geben Sie dem Öffnen-Button 8 den Instanznamen »open_btn«. Dem Schließen-Button 9 geben Sie den Instanznamen »close_btn«. Mit diesen Buttons wird der User dem Player später MP3-Tracks hinzufügen und das Fenster schließen können.



◀ **Abbildung 4**

Dem Öffnen-Button wird der Instanzname »open_btn« gegeben.



◀ **Abbildung 5**

Dem Schließen-Button wird der Instanzname »close_btn« gegeben.

Wählen Sie den Hintergrund in der Ebene »bg«, und konvertieren Sie ihn über **MODIFIZIEREN • IN SYMBOL KONVERTIEREN...** in eine Schaltfläche mit dem Namen »bg«. Als Instanznamen vergeben Sie »bg_btn«. Später kann der User, wenn er auf den Hintergrund des Players klickt und die Maustaste gedrückt hält, die Position des Players verschieben.



Schritt 5

Hintergrund in Button umwandeln

◀ **Abbildung 6**

Der Hintergrund wird in ein Symbol umgewandelt.



◀ **Abbildung 7**

Dem Hintergrund wird der Instanzname »bg_btn« gegeben.

Nun fügen Sie der Dokumentklasse die zusätzlich benötigten Imports und Variablen hinzu. Öffnen Sie dafür die Datei **PLAYER.AS** aus dem Workshop »Soundspektrum«.

Schritt 6

Imports und Variablen hinzufügen

Zwischenergebnis:
32_AIR-MUSIKPLAYER
• PLAYER_STEP06.AS



In Zeile 16 ergänzen Sie folgende Import-Anweisung:

```
16:     import flash.filesystem.File;
```

Außerdem deklarieren Sie in Zeile 36 folgende Variable:

```
36:     private var openFile:File;
```

Schritt 7

Button-Events anmelden

Zwischenergebnis:
32_AIR-MUSIKPLAYER
• PLAYER_STEP07.AS



Fügen Sie ab Zeile 119 folgenden Code hinzu:

```
119:     bg_btn.addEventListener(MouseEvent.MOUSE_DOWN,  
        onDragPress);  
120:     close_btn.addEventListener(MouseEvent.CLICK,  
        onCloseRelease);  
121:     open_btn.addEventListener(MouseEvent.CLICK,  
        onUploadRelease);
```

Wenn der User jetzt auf die jeweiligen Buttons klickt, wird die als zweiter Parameter übergebene Methode ausgeführt.

Schritt 8

Button-Funktionen anlegen

Zwischenergebnis:
32_AIR-MUSIKPLAYER
• PLAYER_STEP08.AS



Jetzt erstellen Sie die Funktionen, die die Userklicks verarbeiten. Ergänzen Sie Ihr Skript ab Zeile 191 wie folgt:

```
191:     private function onDragPress(evt:MouseEvent):void  
192:     {  
193:     }  
194:     private function onCloseRelease(evt:MouseEvent):void  
195:     {  
196:     }  
197:     private function onUploadRelease(evt:MouseEvent):void  
198:     {  
199:     }
```

Schritt 9

Drag-Funktion anlegen

Fügen Sie folgende Zeile als Zeile 193 ein:

```
193:     stage.nativeWindow.startMove();
```

Testen Sie Ihr Dokument über **STEUERUNG • FILM TESTEN** oder `[Strg]/[⌘]+[↵]`. Wenn Sie jetzt auf dem Hintergrund die Maustaste gedrückt halten, können Sie Ihre Anwendung über den Bildschirm ziehen.



Zwischenergebnis:
32_AIR-MUSIKPLAYER
• PLAYER_STEP09.AS

Ergänzen Sie Ihren Code in Zeile 197 wie folgt:

```
197: stage.nativeWindow.close();
```

Testen Sie Ihren Film nun über **STEUERUNG • FILM TESTEN** oder `[Strg]/[⌘]+[↵]`. Bei einem Klick auf den Schließen-Button wird die Anwendung geschlossen.



Zwischenergebnis:
32_AIR-MUSIKPLAYER
• PLAYER_STEP10.AS

Mit einem Klick auf den Öffnen-Button soll der User eigene Musikstücke hinzufügen können. Ergänzen Sie Ihren Code um folgende Zeilen:

```
201: openFile = File.desktopDirectory;  
202: openFile.browseForOpen("Add mp3...");  
203: openFile.addEventListener(Event.SELECT, onSelectFile);
```

In Zeile 201 erstellen Sie zunächst eine Referenz zur Datei, die von dem User hinzugefügt wird. `File.desktopDirectory` bewirkt, dass automatisch das DESKTOP-Verzeichnis im Öffnen-Dialog angezeigt wird.

Der Dialog selbst wird in Zeile 202 geöffnet. Der Text »Add mp3...« steht in der Titelleiste des Dialogfelds. Um die Wahl des Users verarbeiten zu können, benötigen Sie ein weiteres Event. In Zeile 203 wird, wenn der User eine Datei ausgewählt hat, die Funktion `onSelectFile()` aufgerufen.



Zwischenergebnis:
32_AIR-MUSIKPLAYER
• PLAYER_STEP11.AS

Die Verarbeitung der Datei findet in der Funktion `onSelectFile()` statt. Ergänzen Sie Ihren Code wie folgt:

```
205: private function onSelectFile(e:Event):void  
206: {  
207:     var currentFile:File = File(e.target);  
208:     if(currentFile.extension == "mp3")  
209:     {  
210:         songs.push(currentFile.url);  
211:     }  
212: }
```



Zwischenergebnis:
32_AIR-MUSIKPLAYER
• PLAYER_STEP12.AS

Schritt 10

Schließen-Funktion hinzufügen

Schritt 11

Öffnen-Funktion hinzufügen

Schritt 12

Geöffnete Datei verarbeiten

In Zeile 207 erstellen Sie eine temporäre Referenz der gewählten Datei in der Variablen `currentFile`.

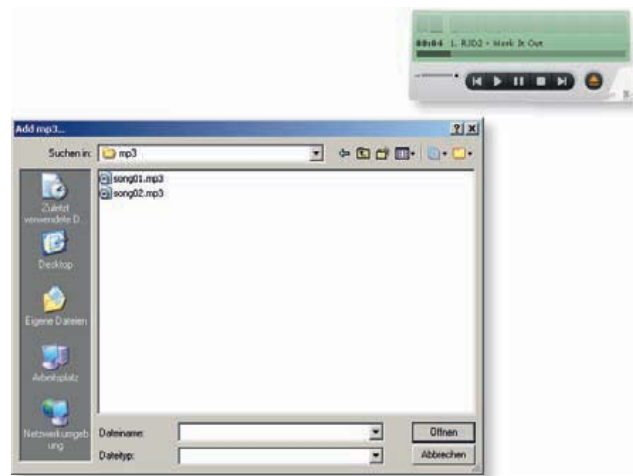
Um sicherzustellen, dass es sich bei der Datei auch um eine MP3-Datei handelt, wird die `extension` der Datei in Zeile 208 abgefragt. Mit Hilfe der `extension`-Eigenschaft können Sie die Dateierweiterung einer zu ladenden Datei prüfen. Eine `jpg`-Datei würden Sie also mit `jpg` abfragen, eine Word-Datei mit `doc`.

In Zeile 210 schließlich wird die Datei dem `songs`-Array hinzugefügt.

Wenn Sie den Film nun testen, können Sie die Anwendung mit einem Klick auf den Schließen-Button schließen und über den Öffnen-Button weitere Musikstücke hinzufügen.

Abbildung 8 ►

Über den Dateibrowser des Betriebssystems können neue Musikstücke hinzugefügt werden. Auf dem Mac öffnet sich der Finder.



Um aus Ihrem Dokument jetzt eine Adobe AIR-Anwendung zu erstellen, rufen Sie **BEFEHLE • AIR - PACKAGE AIR FILE** auf. Die AIR-Datei mit dem Namen `AIRMUSICPLAYER.AIR` befindet sich nun in demselben Ordner wie Ihre `fla`-Datei.

Die fertige AIR-Datei können Sie (nachdem Sie eine korrekte Signatur angelegt haben) Ihren Freunden mailen oder auf Ihrer Website zum Download anbieten.

Abbildung 9 ►

Ihre erste AIR-Anwendung ist fertig.



Video-Player

Flash-Videos im eigenen Player abspielen

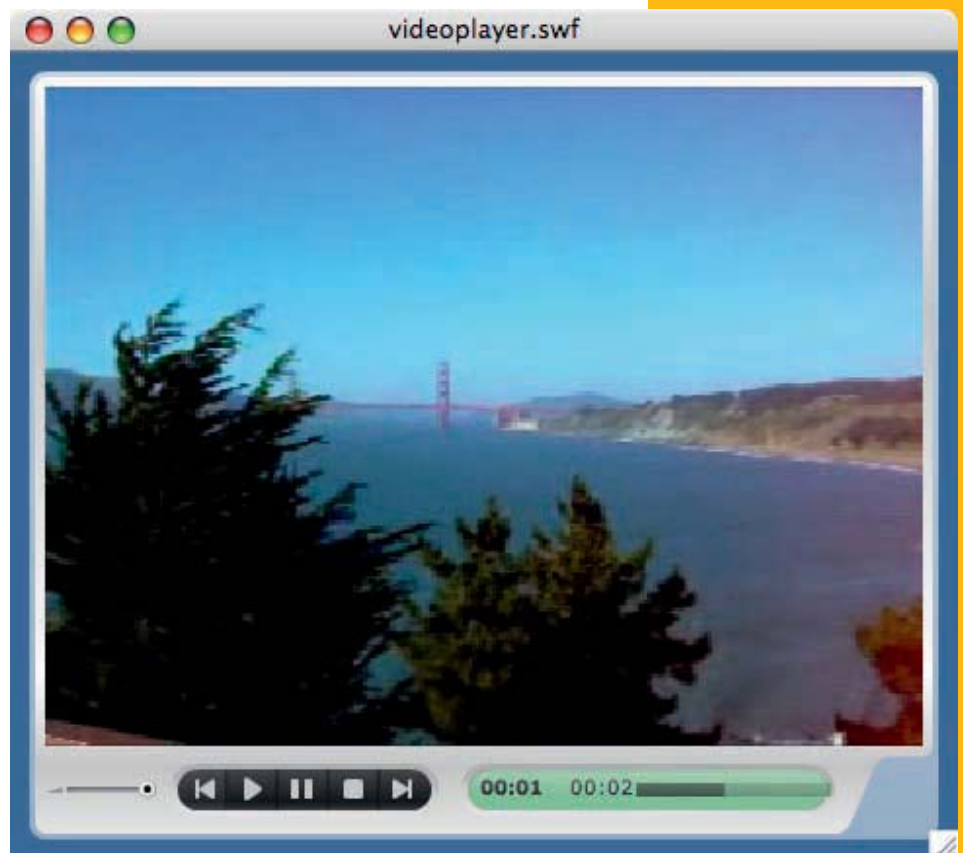
Das Flash-Video-Format, kurz flv, ist im Internet sehr weit verbreitet. Es gibt zahlreiche Video-Portale, die mit dem flv-Format arbeiten. Als Beispiel kann man eins der größten nennen: youtube. Flash bietet mit seinen Komponenten eine Reihe von Möglichkeiten, Videos abzuspielen. Jedoch sind diese Möglichkeiten meist an ein festes oder unflexibles Layout gebunden. In diesem Workshop werden Sie sehen, wie einfach es ist, einen eigenen Video-Player mit ActionScript 3 zu programmieren.

Zielsetzungen:

- Video abspielen
- Videosteuerung programmieren
- Ladefortschritt anzeigen
- Abspielposition anzeigen

[Ergebnis: 33_VIDEO-PLAYER • VIDEOPLAYER.SWF]

```
package
{
    import flash.display.MovieClip;
    import flash.display.SimpleButton;
    import flash.text.TextField;
    import fl.video.VideoPlayer;
    import flash.events.MouseEvent;
    import fl.video.VideoEvent;
    import fl.video.VideoProgressEvent;
    import flash.geom.Rectangle;
    public class Player extends MovieClip
```



Schritt 1 Vorbereitungen

Im ersten Schritt legen Sie einen neuen Ordner auf Ihrer Festplatte an und nennen ihn VIDEOPLAYER. Hier werden Sie die von Ihnen erstellten Beispiele zu diesem Workshop abspeichern.

Schritt 2 Bedienelemente

Die Bedienelemente des Players müssen Sie nicht selbst erstellen und anordnen. Denn auf der Buch-DVD im Ordner 33_VIDEO-PLAYER finden Sie die Flash-Datei VIDEOPLAYER_TEMPLATE.FLA, in der die Bedienelemente bereits vorhanden sind. Kopieren Sie die Datei in den Projektordner auf Ihrer Festplatte, und benennen Sie sie in VIDEOPLAYER.FLA um.

Wenn Sie die Datei öffnen, sehen Sie, dass sich Buttons für Play ④, Pause ⑤, Stopp ⑥, Vor ⑦ und Zurück ③ auf der Bühne befinden. Des Weiteren sind Movieclips für einen Lautstärkeregler ②, die Bildfläche ① sowie zwei Statusbalken ⑩ und ⑪ vorhanden. Außerdem sehen Sie zwei Textfelder ⑧ und ⑨, die später zur Zeitanzeige eingesetzt werden.

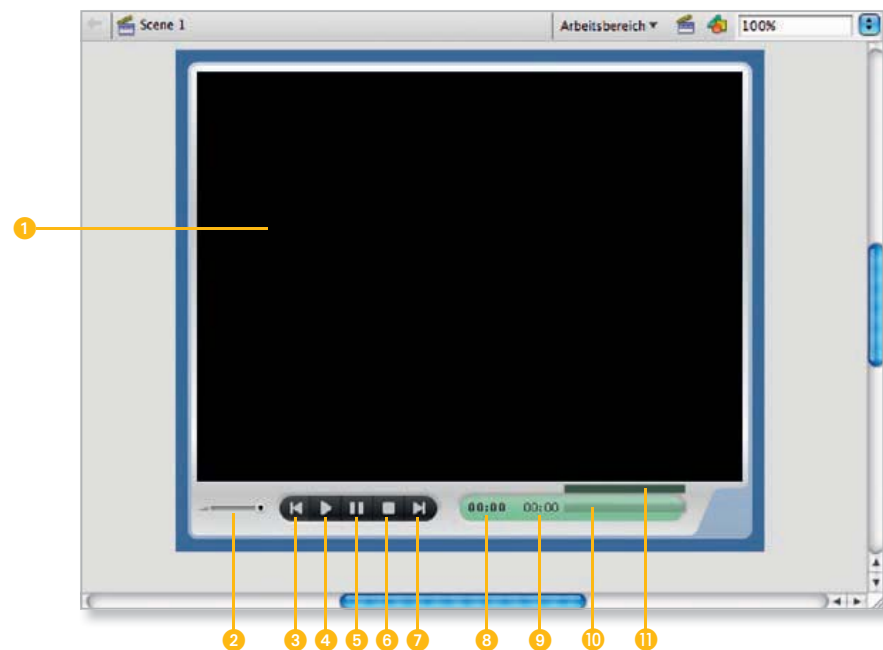
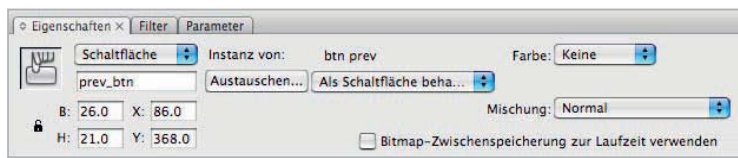


Abbildung 1 ►
Die Oberfläche des
Players haben wir für Sie
bereits erstellt.

Schritt 3 Instanznamen der Bedienelemente

Bei der Programmierung des Players werden die Bedienelemente über die Instanznamen angesprochen, um ihnen ihre Aktionen zuzuweisen. Markieren Sie die Abspielbuttons nacheinander von links nach rechts, und tragen Sie im Eigenschaftsfenster folgende Instanznamen der Reihe nach ein: »prev_btn«, »play_btn«, »pause_btn«, »stop_btn« und »next_btn«.

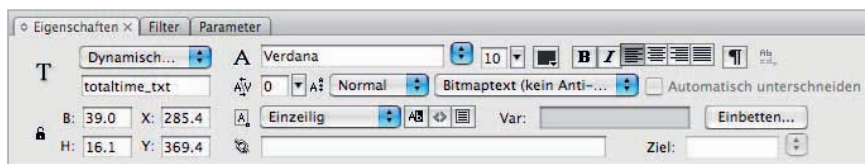


◀ **Abbildung 2**

Instanzname des Movieclips
»prev_btn«

Der Lautstärkeregler erhält den Namen »sliderVolume_mc«.

Der obere Statusbalken 11 soll den Abspielfortschritt anzeigen. Geben Sie ihm den Namen »positionBar_mc«. Der andere Balken 10 soll den Lade fortschritt des Videos anzeigen. Weisen Sie diesem Balken den Namen »loadingBar_mc« zu. Die Bildfläche 1 erhält den Instanznamen »screen_mc«. Das linke Textfeld 8 soll die Abspielzeit anzeigen. Geben Sie ihm den Namen »time_txt«. Das zweite Textfeld 9 wird die Gesamtdauer des Films anzeigen. Benennen Sie es mit »totaltime_txt«.



◀ **Abbildung 3**

Instanzname des zweiten
Textfelds eintragen

Damit der Abspielfortschritt und der Ladestatus übersichtlicher angezeigt werden und das Verhältnis der vorgeladenen Datenmenge zu der bereits abgespielten Datenmenge klar wird, verschieben Sie nun `positionBar_mc` auf die Position von `loadingBar_mc`.



Achtung

Sollten Sie es vorziehen, eigene Bedienelemente zu kreieren, so achten Sie darauf, die Instanznamen der Bedienelemente wie oben angegeben zuzuweisen.

◀ **Abbildung 4**

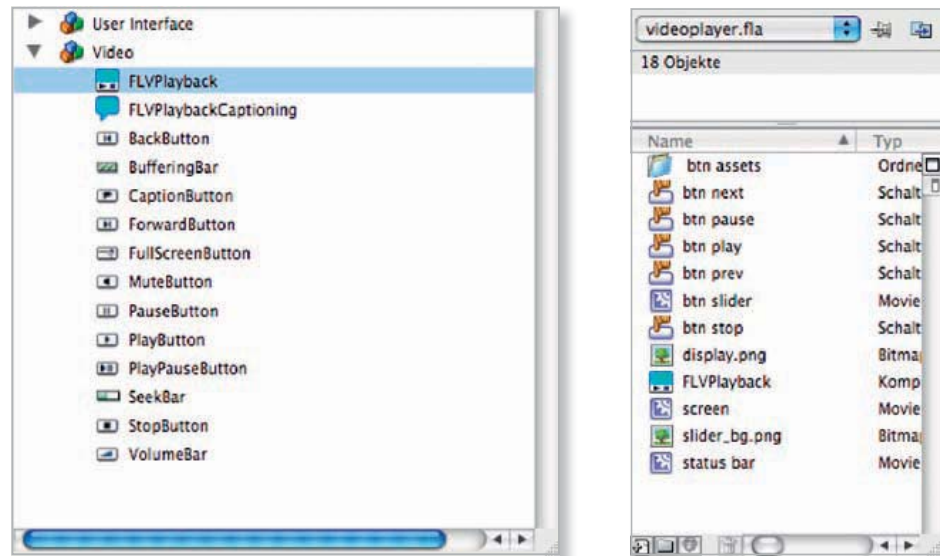
Ladebalken positionieren

Flash besitzt eine `VideoPlayer`-Klasse, die sich in dem Package `fl.video` befindet. In der Flash-Hilfe können Sie sich über den Funktionsgehalt dieser Klasse informieren. Damit Sie die Vorzüge der Klasse für Ihren eigenen Video-Player nutzen können, ist es notwendig, eine Instanz der Komponente `FLVPlayback` in der Bibliothek zu haben. Erst dann steht Ihnen die Klasse `fl.video.VideoPlayer` zur Verfügung. Öffnen Sie das **KOMPONENTEN**-Fenster mit `[Strg]/[⌘] + [F7]`. Klappen Sie den Eintrag **VIDEO** auf, und ziehen Sie **FLVPLAYBACK** in die Bibliothek.

Schritt 4

FLVPlayback-Komponente einfügen

Abbildung 5 ►
Per Drag & Drop gelangt die
FLVPlayback-Komponente
in die Bibliothek Ihres Flash-
Films.



Schritt 5 Anlegen der Dokument- klasse

Sie haben jetzt alle Vorbereitungen vorgenommen, um mit der Programmierung beginnen zu können. Zu diesem Zweck legen Sie über `[Strg]/[N]` eine neue ACTIONSCRIPT-DATEI an. Speichern Sie diese Klasse unter dem Namen `PLAYER.AS` im Projektordner.

In der neu angelegten Datei werden Sie in den nächsten Schritten dieses Workshops alle Funktionen des Players umsetzen. Fangen Sie mit dem grundlegenden Klassengerüst an:

```
1: package
2: {
3:     import flash.display.MovieClip;
4:     import flash.display.SimpleButton;
5:     import flash.text.TextField;
6:     import fl.video.VideoPlayer;
7:     public class Player extends MovieClip
8:     {
9:         public function Player():void
10:        {
11:        }
12:    }
13: }
```

Die Dokumentklasse muss in diesem Fall von `MovieClip` oder `Sprite` erben, da diese hier im Grunde die Funktionalität von `root` übernimmt. In Zeile 3 importieren Sie zu diesem Zweck die `MovieClip`-Klasse. Weil auch bereits

Schaltflächen auf der Bühne liegen, stellen Sie in der darauffolgenden Zeile zusätzlich die Klasse `flash.display.SimpleButton` per Import zur Verfügung. Da sich ebenfalls Textfelder auf der Bühne befinden, importieren Sie auch die Klasse `flash.text.TextField`. Des Weiteren importieren Sie die Klasse `fl.video.VideoPlayer`, um diese nutzen zu können.

Wechseln Sie zurück zur Datei `VIDEOPLAYER.FLA`, und weisen Sie der Datei innerhalb des Eigenschaftensfensters »Player« als DOKUMENTKLASSE zu. Bestätigen Sie mit .



◀ **Abbildung 6**
Die Dokumentklasse
zuweisen

Deklaren bzw. initialisieren Sie nun folgende Klassenvariablen über dem Konstruktor in der Player-Klasse:

```
private var videoPlayer:VideoPlayer;
private var videos:Array = ['http://www.flashfastforward.de/
videos/video1.flv','http://www.flashfastforward.de/videos/video2.
flv'];
private var currentVideo:uint;
```

Als Erstes deklarieren Sie die Variable `videoPlayer`. In ihr wird eine Instanz der Klasse `fl.video.VideoPlayer` gespeichert. Das in der nächsten Zeile deklarierte Array `videos` wird mit Strings gefüllt, die die URLs zu den Video-Dateien festlegen. Die in der letzten Zeile deklarierte Variable `currentVideo` wird als Index dienen, um die in `videos` gespeicherten URLs zu referenzieren.

Alles, was Sie tun müssen, um nun ein Video abspielen zu können, ist, eine Instanz der `VideoPlayer`-Klasse zu erzeugen und deren `play`-Methode aufzurufen. Da die `VideoPlayer`-Klasse von `Sprite` erbt, können Sie die Instanz ganz einfach per `addChild()` auf die Bühne bringen und das Video wird angezeigt.

Programmieren Sie eine Methode `initVideoPlayer()`, in der Sie die `VideoPlayer`-Instanz erzeugen. Fügen Sie folgenden Code ans Ende Ihrer Klasse ein.

Schritt 6

Erste Klassenvariablen

Schritt 7

Video abspielen

```

1: private function initVideoPlayer():void
2: {
3:     videoPlayer = new VideoPlayer();
4:     videoPlayer.setSize(screen_mc.width, screen_mc.height);
5:     screen_mc.addChild(videoPlayer);
6: }

```

In Zeile 3 erzeugen und speichern Sie in der Klassenvariable `videoPlayer` eine Instanz der Klasse `VideoPlayer`. In der nächsten Zeile verwenden Sie die Methode `setSize()`, um das Format des Videos zu bestimmen. Als Werte übergeben Sie die Breite und die Höhe von `screen_mc`. Das Video wird damit an die Maße von `screen_mc` angepasst. Danach, in Zeile 5, fügen Sie `screen_mc` das Video als Kindelement hinzu. Es befindet sich damit genau dort, wo Sie es haben wollen: auf dem Bildschirm Ihres Video-Players.

Was jetzt noch fehlt, um ein Video abzuspielen, sind der Aufruf der soeben programmierten Methode und der Aufruf von `play()`. Erledigen Sie beides folgendermaßen im Konstruktor:

```

initVideoPlayer();
videoPlayer.play(videos[currentVideo]);

```

In der ersten Zeile rufen Sie die Methode `initVideoPlayer()` auf. Die `VideoPlayer`-Instanz ist damit erzeugt und in `videoPlayer` gespeichert. In der zweiten Zeile rufen Sie deren `play`-Methode auf und übergeben dieser die URL zu dem ersten Video. Da die Variable `currentVideo` vom Typ `uint` ohne Initialisierung automatisch 0 ist, verweist sie auf die erste Stelle des Arrays `videos`.

Schritt 8

Film testen

Sie werden es kaum glauben, aber das war schon alles, was Sie tun müssen, um ein Video abzuspielen. Testen Sie nun Ihren Flash-Film mit `Strg`/`⌘` + `↵`, und Sie werden es sehen!

Natürlich funktionieren die Bedienelemente noch nicht – auch wenn Sie zehnmal den Pause-Button anklicken. Darum werden wir uns in den nächsten Schritten kümmern.



◀ **Abbildung 7**
Das Video wird abgespielt.

Um festzustellen, ob auf einen Button geklickt wurde, und dann eine Aktion auszuführen, benötigen Sie einen Maus-Listener. Importieren Sie dazu als Erstes die `MouseEvent`-Klasse unter der letzten Import-Anweisung:

```
import flash.events.MouseEvent;
```

Programmieren Sie jetzt die Methode `initControls()`, in der Sie jedem Button einen Maus-EventListener hinzufügen, der bei jedem Klick eine Methode `onClick()` aufruft. Fügen Sie den Code unter die in Schritt 7 erstellte Methode `initVideoPlayer()` ein:

```
private function initControls():void
{
    play_btn.addEventListener(MouseEvent.CLICK, onClick);
    pause_btn.addEventListener(MouseEvent.CLICK, onClick);
    stop_btn.addEventListener(MouseEvent.CLICK, onClick);
    prev_btn.addEventListener(MouseEvent.CLICK, onClick);
    next_btn.addEventListener(MouseEvent.CLICK, onClick);
}
```

Damit dies auch alles geschieht, müssen Sie noch den Methodenaufruf von `initControls()` einbauen. Schreiben Sie dazu im Konstruktor direkt unter `initVideoPlayer()`; den Aufruf:

```
initControls();
```

Schritt 9

Buttonklick abfragen

Schritt 10

Buttonaktionen programmieren

Bei jedem Buttonklick wird nun also die Methode `onClick()` aufgerufen. In dieser gilt es nun, innerhalb einer `switch`-Anweisung abzufragen, welcher Button angeklickt wurde, um dann dementsprechend `videoPlayer` zu steuern. Programmieren Sie die Methode folgendermaßen unterhalb der letzten Methode Ihrer Klasse:

```
private function onClick(event : MouseEvent):void
{
    switch(event.target){
        case play_btn:
            videoPlayer.play();
            break;
        case pause_btn:
            videoPlayer.pause();
            break;
        case stop_btn:
            videoPlayer.stop();
            videoPlayer.seek(0);
            break;
    }
}
```

Wenn auf `play_btn` geklickt wird, rufen Sie die `play`-Methode von `videoPlayer` auf. Ohne Parameterübergabe hat dies zur Folge, dass das Video, falls es gestoppt ist oder pausiert, weiter abspielt. Im Falle, dass auf `pause_btn` geklickt wurde, wird das Video durch den Aufruf von `pause()` zum Pausieren gebracht. Wurde auf `stop_btn` geklickt, wird die `stop`-Methode von `videoPlayer` aufgerufen. Diese führt ohne weitere Einstellungen nichts anderes als `pause()` aus. Deswegen rufen Sie außerdem noch die `seek`-Methode von `videoPlayer` auf und übergeben dieser den Wert 0. Das Video wird damit an den Anfang zurückgespult.

Schritt 11

Film testen

Testen Sie nun Ihren Film, und probieren Sie die Buttons aus. Es funktioniert! Sie können das Video pausieren lassen, wieder abspielen und stoppen.

Abbildung 8 ►

Die Funktionalität für die Play-, Pause- und Stopp-Buttons wurde erstellt.



Was noch fehlt, sind die Aktionen der Vor- und Zurück-Buttons. Diese realisieren Sie in den Methoden `playNext()` und `playPrev()`, die Sie folgendermaßen, erneut unterhalb der letzten Methode Ihres Skripts, programmieren:

```
private function playNext():void
{
    currentVideo = currentVideo >= videos.length-1 ? 0 :
    ++currentVideo;
    videoPlayer.play(videos[currentVideo]);
}
```

In der `playNext`-Methode inkrementieren Sie `currentVideo`. In dem Fall, dass `currentVideo` größer als die Länge des Arrays `videos` ist, setzen Sie `currentVideo` wieder auf 0. Das Spiel kann damit von vorn beginnen.

```
private function playPrev():void
{
    currentVideo = currentVideo <= 0 ? videos.length-1 :
    --currentVideo;
    videoPlayer.play(videos[currentVideo]);
}
```

In der `playPrev`-Methode dekrementieren Sie `currentVideo` solange, bis diese Variable den Wert 0 erreicht. Danach wird der Wert auf `videos.length-1` gesetzt, also auf den Index des letzten Elements des Arrays `videos`.

Damit diese Methoden auch aufgerufen werden, erweitern Sie nun die `switch`-Anweisung in der `onClick`-Methode um zwei weitere `case`-Bedingungen:

```
case next_btn:
    playNext();
    break;
case prev_btn:
    playPrev();
    break;
```

Testen Sie nun wieder Ihren Film, und probieren Sie die Vor- und Zurück-Buttons aus. Da Sie nur URLs zu zwei Videos in Ihrem `videos`-Array gespeichert haben, wird bei einem Klick auf die Buttons immer zwischen den

Schritt 12

Vor- und Zurück-Buttons

Schritt 13

Film testen

beiden Videos gewechselt. Das jedoch beweist umso mehr, dass die kleine Endlosschleife auch richtig funktioniert, denn es wird niemals ein falscher Wert im Array adressiert.

Abbildung 9 ►

Über die Vor- und Zurück-Buttons können Sie zwischen den beiden Videos wechseln.



Schritt 14 Abspielposition anzeigen

Um die Abspielposition abzufragen, bietet die `VideoPlayer`-Klasse ein Event, dem ein Listener zugefügt werden muss. Importieren Sie dazu oben in den Import-Anweisungen der Klasse die Klasse `VideoEvent`:

```
import fl.video.VideoEvent;
```

Fügen Sie nun `videoPlayer` in der letzten Zeile der `initVideoPlayer`-Methode einen Listener hinzu, der eine Methode `onPlayheadUpdate()` aufruft:

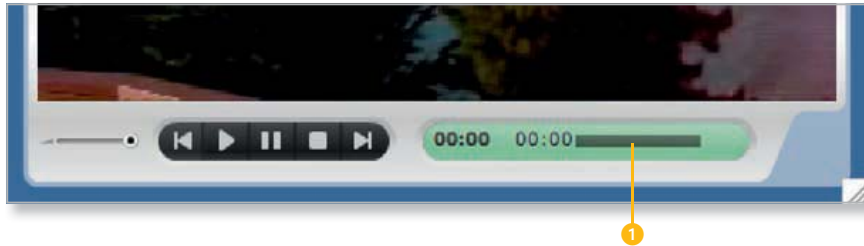
```
videoPlayer.addEventListener(VideoEvent.PLAYHEAD_UPDATE,  
    onPlayheadUpdate);
```

Programmieren Sie nun die `onPlayheadUpdate`-Methode folgendermaßen am Ende Ihres Skripts:

```
private function onPlayheadUpdate(event:VideoEvent):void  
{  
    var progress:Number = event.playheadTime / videoPlayer.  
        metadata.duration;  
    positionBar_mc.scaleX = progress;  
}
```

Das Event-Objekt `event` besitzt die Eigenschaft `playHeadTime`, in der die abgespielten Sekunden gespeichert sind. Diesen Wert teilen Sie durch die Gesamtspielzeit des Videos, die in der `duration`-Eigenschaft des `metadata`-Objekts von `videoPlayer` ebenfalls in Sekunden gespeichert ist. Den errechneten Wert, der zwischen 0 und 1 liegt, weisen Sie der `scaleX`-Eigenschaft des `positionBar_mc`-Movieclips zu.

Testen Sie nun wieder Ihren Film. Wie Sie sehen, wird der Statusbalken zu Beginn des Films auf minimale Größe skaliert und vergrößert sich parallel zur Abspielposition des Videos, bis er am Ende seine volle Länge erreicht hat.



Schritt 15

Film testen

◀ Abbildung 10

Der Statusbalken 1 zeigt die aktuelle Position im Video.

Den Ladefortschritt können Sie ebenfalls mit einem Listener abfragen. Importieren Sie dazu unter den bisherigen Import-Anweisungen die Klasse `VideoProgressEvent`:

```
import fl.video.VideoProgressEvent;
```

Fügen Sie nun `videoPlayer` in der letzten Zeile der `initVideoPlayer`-Methode einen Listener hinzu, der eine Methode `onProgress()` aufruft:

```
videoPlayer.addEventListener(VideoProgressEvent.PROGRESS,  
    onProgress);
```

Programmieren Sie anschließend die `onProgress`-Methode folgendermaßen am Ende des Skripts unter der letzten Methode:

```
private function onProgress(event:VideoProgressEvent):void  
{  
    var progress:Number = event.bytesLoaded/event.bytesTotal  
    loadingBar_mc.scaleX = progress;  
}
```

Das Event-Objekt `event` besitzt die Eigenschaften `bytesLoaded` und `bytesTotal`. Die Eigenschaft `bytesLoaded` speichert die Größe des geladenen Inhalts in Bytes und wird durch `bytesTotal` geteilt. In der Variablen `progress` wird dann eine Zahl zwischen 0 und 1 gespeichert, die Sie anschließend der `scaleX`-Eigenschaft von `loadingBar_mc` zuweisen.

Schritt 16

Ladefortschritt anzeigen

Schritt 17

Film testen

Testen Sie nun wieder Ihren Film. Abhängig von der Bandbreite Ihrer Internetverbindung können Sie sehen, dass nun beide Statusbalken zu Beginn des Films auf minimale Größe skaliert werden. Der Ladebalken vergrößert sich parallel zum Ladefortschritt des Videos, bis er am Ende seine volle Länge erreicht hat.

Abbildung 11 ►

Der Ladebalken ❶ vergrößert sich entsprechend dem Ladefortschritt.



Schritt 18

Die Abspielzeit anzeigen

Nachdem Sie – in Schritt 14 – die Abspieldauer und die Abspielzeit abgefragt haben, können Sie jetzt die Zeitanzeige aktualisieren. Die Zeit wird in Sekunden geliefert. Der Wert ist jetzt in einen String umzuwandeln, damit er in das Format »00:00« gebracht werden kann. Schreiben Sie dazu unter der letzten Methode eine weitere Methode `getTimeString()`, der Sie die Sekunden übergeben können und die einen String zurückliefert:

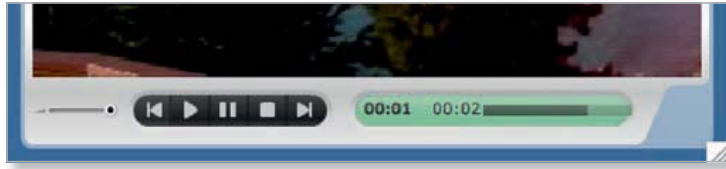
```
1: private function getTimeString(seconds:Number):String
2: {
3:     var min:int = Math.floor(seconds/60);
4:     var sec:int = Math.floor(seconds)%60;
5:     var mString:String = (min < 10) ? "0"+min : min.
        toString();
6:     var sString:String = (sec < 10) ? "0"+sec : sec.
        toString();
7:     return mString + ":" + sString;
8: }
```

In Zeile 3 errechnen Sie die Minuten, indem Sie die übergebenen `seconds` durch 60 teilen. In der nächsten Zeile ermitteln Sie die restlichen Sekunden. In den Zeilen 5 und 6 stellen Sie sicher, dass jeweils eine Null vor das Ergebnis gestellt wird, falls die Zahlen kleiner als zehn sind. Schließlich geben Sie den formatierten String zurück.

Erweitern Sie nun die `onPlayheadUpdate`-Methode um folgende zwei Zeilen, um die Zeiten in den Textfeldern `time_txt` und `totaltime_txt` anzuzeigen:


```
time_txt.text = getFormattedTime(event.playheadTime);
totaltime_txt.text = getFormattedTime(videoPlayer.metadata.
duration);
```

Testen Sie Ihren Film. Wie Sie sehen, wird die Zeit aktualisiert, sobald das Video abgespielt wird.



Schritt 19

Film testen

◀ **Abbildung 12**
Die Zeit wird aktualisiert.

Wenn Sie sich den ersten Film vollständig angesehen haben, werden Sie feststellen, dass der Video-Player einfach stoppt. Das entspricht natürlich nicht unseren Vorstellungen, denn in diesem Fall soll direkt das nächste Video abgespielt werden. Deswegen müssen Sie einen Listener anfügen, der auf das Event lauscht, dass das Video am Ende angekommen ist. Hierbei handelt es sich um das `COMPLETE`-Event der `VideoEvent`-Klasse.

Diesem Event fügen Sie in der `initVideoPlayer`-Methode einen Listener hinzu, der die Methode `playNext()` aufruft, damit bei Video-Ende das nächste Video in der Liste abgespielt wird. Ergänzen Sie folgenden Code in der letzten Zeile der `initVideoPlayer`-Methode:

```
videoPlayer.addEventListener(VideoEvent.COMPLETE, playNext);
```

Da die `playNext`-Methode nun eine Listener-Methode ist, erwartet sie einen Parameter: das Event-Objekt. Ändern Sie also die Deklaration von `playNext()` wie folgt:

```
private function playNext(event:VideoEvent = null):void
```

Sie weisen `event` den Wert `null` zu, damit keine Fehlermeldungen kommen, wenn Sie die Methode ohne Parameter aufrufen.

Testen Sie nun wieder Ihren Film. Wenn Sie sich nicht den ganzen Film anschauen möchten, können Sie diesen Schritt auch ans Ende des Workshops verlegen. Wenn Sie das Video jedoch bis zu seinem Ende durchlaufen lassen, werden Sie sehen, dass dann das nächste Video abgespielt wird.

Schritt 20

Endlosschleife

Schritt 21

Film testen

Schritt 22

Lautstärkeregler vorbereiten

startdrag und rectangle

Sie können mit der `Rectangle`-Klasse ein Rechteck erzeugen, das den Begrenzungsrahmen bestimmt, in dem ein Movieclip oder ein Sprite per `startDrag()` bewegt werden kann.

Um die Lautstärke zu regeln, muss der Lautstärkeregler `sliderVolume_mc` in einem bestimmten Bereich bewegbar sein. Hierfür nutzen Sie die Vorteile der Methoden `startDrag()` und `stopDrag()` in Verbindung mit der `Rectangle`-Klasse.

Importieren Sie diese Klasse:

```
import flash.geom.Rectangle;
```

Deklarieren Sie nun im Deklarationsbereich der Klasse eine Variable `volumeRect`, in der Sie eine Instanz von `Rectangle` speichern werden:

```
private var volumeRect:Rectangle;
```

Initialisieren Sie diese Variable in der letzten Zeile der `initControls`-Methode:

```
volumeRect = new Rectangle(sliderVolume_mc.x-40,  
    sliderVolume_mc.y,40,0);
```

Mit dieser Zeile weisen Sie `volumeRect` eine Instanz von `Rectangle` zu. Die x-Position des Rechtecks setzen Sie auf die x-Position von `sliderVolume_mc` abzüglich 40 Pixel. Die y-Position setzen Sie auf die von `sliderVolume_mc`. Die Breite bestimmen Sie mit 40 Pixel, also dem Wert, den Sie eben von der x-Position abgezogen haben. Der Höhe geben Sie 0 Pixel. Es ist nun also ein Rechteck von 40 Pixel Breite entstanden, das 40 Pixel vor `sliderVolume_mc` beginnt und an der x-Position von `sliderVolume_mc` endet.

Ziel ist es nun, der `startDrag`-Methode dieses Rechteck als Beschränkung für das Verschieben von `sliderVolume_mc` zu übergeben. `sliderVolume_mc` soll sich von seiner x-Position 40 Pixel nach links und wieder zurück bewegen lassen. Fügen Sie zu diesem Zweck in der nächsten Zeile `sliderVolume_mc` einen Maus-Listener hinzu, der bei einem Klick eine Methode `dragVolume()` aufruft:

```
sliderVolume_mc.addEventListener(MouseEvent.CLICK,  
    dragVolume);
```

Schritt 23

Lautstärkeregler programmieren

Programmieren Sie nun die `dragVolume`-Methode, die bei einem Klick auf den Lautstärkeregler ausgelöst wird. Ergänzen Sie unten in Ihrer Klasse unter der letzten Methode folgenden Code:

```

1: private function dragVolume(event:MouseEvent):void
2: {
3:     sliderVolume_mc.startDrag(false, volumeRect);
4:     stage.addEventListener(MouseEvent.MOUSE_MOVE,
        onDragVolume);
5: }

```

In Zeile 3 rufen Sie die `startDrag`-Methode von `sliderVolume_mc` auf und übergeben ihr `volumeRect` als Begrenzungsrahmen. In Zeile 4 fügen Sie nun `stage` einen Listener hinzu, der bei Bewegung der Maus permanent eine Methode `onDragVolume()` aufruft. In dieser Methode realisieren Sie nun die Funktionalität des Lautstärkereglers. Übernehmen Sie dazu folgenden Code unter die eben geschriebene Funktion:

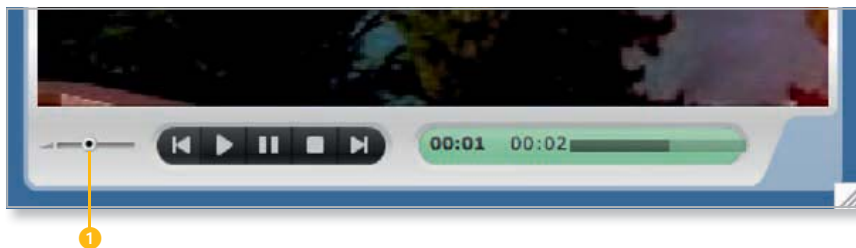
```

6: private function onDragVolume(event:MouseEvent):void
7: {
8:     var volume:Number = (sliderVolume_mc.x - volumeRect.x)/vo-
        lumeRect.width;
9:     videoPlayer.volume = volume;
10: }

```

Sie errechnen in Zeile 8 die Differenz zwischen der x-Position des Lautstärkereglers und der x-Position des Rechtecks und teilen das Ergebnis durch die Breite des Rechtecks. Das Divisionsergebnis speichern Sie dann in der Variablen `volume`. So entsteht ein Wert zwischen 0 und 1, den Sie der `volume`-Eigenschaft von `videoPlayer` zuweisen.

Testen Sie nun erneut Ihren Film. Klicken Sie auf den Lautstärkeregler ①, und bewegen Sie ihn nach links und wieder zurück. Wie Sie feststellen, passt sich die Lautstärke des Videos genau der Position des Reglers an.



Schritt 24

Film testen

◀ **Abbildung 13**
Bewegen Sie mit der Maus den Lautstärkeregler.

Wenn Sie die Maustaste jetzt allerdings wieder loslassen, bewegt sich der Lautstärkeregler weiter mit der Maus und die Lautstärke wird eben-

falls weiterhin angepasst. Das liegt daran, dass Sie bis jetzt noch nicht die `stopDrag`-Methode angewendet haben und der Maus-Listener immer noch aktiv ist. Dies werden Sie im nächsten Schritt beheben.

Schritt 25

Maus-Listener entfernen

Fügen Sie `stage` in der letzten Zeile der `initControls`-Methode einen weiteren Listener hinzu, der beim Loslassen der Maustaste eine Methode `dropVolume()` aufruft:

```
stage.addEventListener(MouseEvent.CLICK, dropVolume);
```

Programmieren Sie nun wie folgt die `dropVolume`-Methode unterhalb der letzten Methode:

```
1: private function dropVolume(event:MouseEvent):void
2: {
3:     sliderVolume_mc.stopDrag();
4:     stage.removeEventListener(MouseEvent.CLICK,
        onDragVolume);
5: }
```

In Zeile 3 rufen Sie die `stopDrag`-Methode von `sliderVolume_mc` auf, was zur Folge hat, dass der Lautstärkeregler nicht weiterhin der Maus folgt. In Zeile 4 entfernen Sie den Listener wieder, der bei jeder Mausbewegung die `onDragVolume`-Methode aufruft.

Schritt 26

Film testen

Testen Sie erneut Ihren Film. Klicken Sie auf den Lautstärkeregler, und bewegen Sie ihn nach links und wieder zurück. Wie Sie feststellen, passt sich die Lautstärke des Videos nach wie vor genau der Position des Lautstärkereglers an. Wenn Sie die Maustaste jetzt loslassen, lassen Sie damit auch den Lautstärkeregler los und die Lautstärke bleibt auf der Position, die Sie eingestellt haben.

Schritt 27

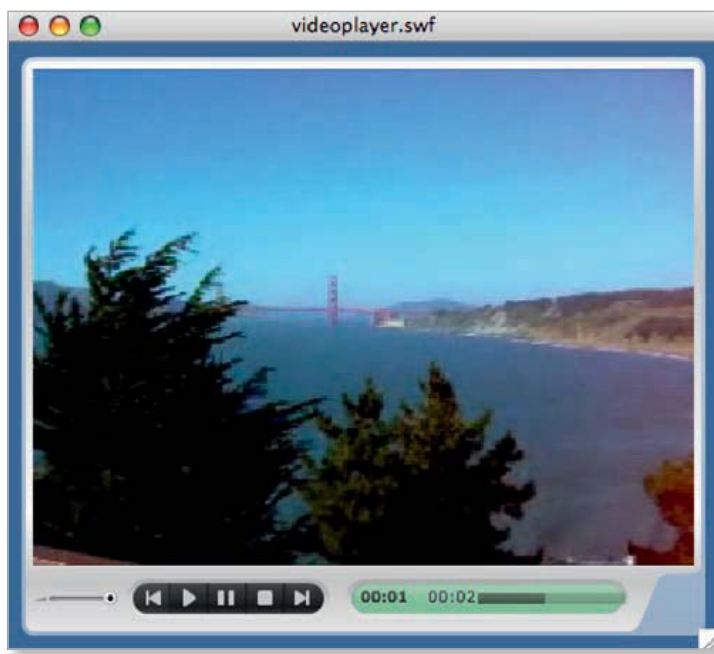
Schönheitsfehler beseitigen

Wie Sie beim Testen des Films feststellen können, sind die beiden Statusbalken, die den Ladefortschritt und die Abspielposition anzeigen, beim Starten des Players auf voller Größe. Das können Sie ganz einfach dadurch ändern, dass Sie in der `initControls`-Methode ganz unten folgende zwei Codezeilen einfügen:

```
positionBar_mc.scaleX = 0;  
loadingBar_mc.scaleX = 0;
```

Sie weisen hier den `scaleX`-Eigenschaften der Statusbalken den Wert 0 zu, wodurch die Balken bei Beginn auf 0 skaliert sind.

Testen Sie erneut Ihren Film. Wie Sie sehen, sind die Statusbalken beim Starten des Films auf 0 skaliert. Herzlichen Glückwunsch, Sie haben soeben einen vollwertigen Video-Player programmiert! Sie können ihn jetzt ganz einfach in Ihre Website oder andere Projekte einbinden.



Schritt 28

Film testen

◀ **Abbildung 14**
Der Player ist fertig und voll funktionsfähig.

Flash im Internet

Flash-Filme in HTML einbinden	406
Zentriert, transparent und in High-Quality	411
Flash Player-Versions-erkennung.....	418
Kommunikation mit HTML	425
Eine Website komplett in Flash	433
Skype-Status	453

Flash-Filme in HTML einbinden

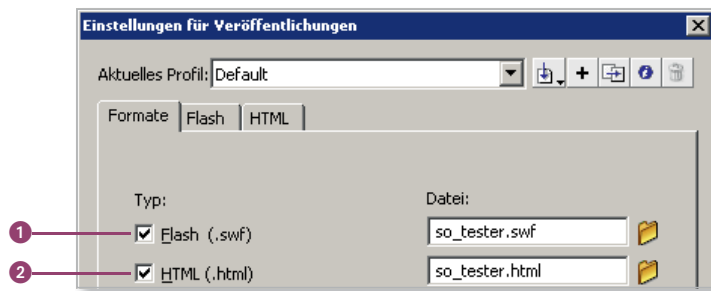
Mit dem SWFObject unnötige Klicks vermeiden

```
<script type="text/javascript" src="swfobject.js"></script>
</head>
<body bgcolor="#000033">
<div id="flashcontent"></div>
<script type="text/javascript">
// 
    var so = new SWFObject("so_tester.swf", "sotest", "300",
        "300", "8.0.24", "#FF6600");
    so.write("flashcontent");
// ]]&gt;
&lt;/script&gt;</pre></div><div data-bbox="55 444 634 810" data-label="Image"><img alt="Screenshot of a web browser displaying a Flash movie titled 'so_tester.html'. The movie shows a sunset scene with a sun, mountains, and a green field. A text box in the center reads 'SWFObject 1.5' and 'this is passed in via FlashVars for example only'. Below it, a note says 'The background color of this flash movie should be set with the flashobject embed to match this box.' followed by a color swatch showing orange and green. The browser window is titled 'so_tester.html - Mozilla Firefox' and the address bar shows 'http://flashfastforward.de/swfobject/'."/></div><div data-bbox="664 222 860 562" data-label="Text"><p><i>In diesem Workshop lernen Sie, wie Sie Ihre fertigen Flash-Filme im Internet mit Hilfe des SWFObject veröffentlichen können – ohne dabei auf lästige Browserspezifikationen achten zu müssen. Mit der neuen Version Flash CS3 hat auch Adobe die HTML-Einbindung von swf-Dateien verbessert. Diese liefert jedoch noch nicht den Komfort bzw. die zusätzlichen Funktionen, die das SWFObject bietet.</i></p></div><div data-bbox="666 606 779 624" data-label="Section-Header"><h2>Zielsetzungen:</h2></div><div data-bbox="676 626 863 665" data-label="List-Group"><ul><li>SWFObject integrieren</li><li>Flash-Film veröffentlichen</li></ul></div><div data-bbox="666 679 835 713" data-label="Text"><p>[Ergebnis: 34_IN-HTML-EINBINDEN]</p></div>
```


Im ersten Schritt legen Sie einen neuen Ordner auf Ihrer Festplatte an und nennen ihn HTML1. Hier werden Sie die Beispiele aus diesem Workshop abspeichern.

Als Vorlage kopieren Sie die Datei SO_TESTER.FLA von der Buch-DVD in den eben erstellten Ordner. Öffnen Sie die Datei dann in der Flash-Entwicklungsumgebung.

Zunächst vergewissern Sie sich über DATEI • EINSTELLUNGEN FÜR VERÖFFENTLICHUNGEN, dass Flash beim Veröffentlichen des Flash-Films zusätzlich zur swf-Datei auch eine html-Datei erzeugt. Im Reiter FORMATE müssen zwei Häkchen gesetzt sein: eines bei FLASH (.SWF) ❶ und ein zweites bei HTML (.HTML) ❷. Lassen Sie die Namen der Dateien unverändert, und bestätigen Sie mit OK.



Wenn Sie nun den Flash-Film mit  + [F12] veröffentlichen, erzeugt Flash im Hintergrund alle benötigten Dateien.

Hier gibt es große Unterschiede zwischen Flash 8 und Flash CS3: In der neuen Version wird zusätzlich zur html-Datei eine JavaScript-Datei angelegt, die sich automatisch um das korrekte Einbetten der swf-Datei in das HTML-Dokument kümmert. Dagegen müssen Sie beim Veröffentlichen mit Flash 8 die in allen Browsern korrekte Integration der swf-Datei selbst implementieren.

Sollten Sie den Internet Explorer als Standardbrowser benutzen, ist Ihnen vielleicht schon einmal aufgefallen, dass ein dünner grauer Rahmen um den Flash-Film erscheint, wenn Sie mit der Maus über die Flash-Inhalte fahren. Zudem taucht nach einigen Sekunden ein Tooltip ❸ auf, der Sie auffordert: »Klicken Sie hier, um dieses Steuerelement zu aktivieren und zu verwenden«.

Schritt 1

Vorbereitungen

Schritt 2

Einstellungen für Veröffentlichungen

◀ Abbildung 1

Im Reiter FORMATE legen Sie fest, welche Formate Flash beim Veröffentlichen anlegen soll.

Schritt 3

Veröffentlichen des Flash-Films



Zwischenergebnis:
34_IN-HTML-EIN-
BINDEN • SO_TESTER_
STEP03.HTML

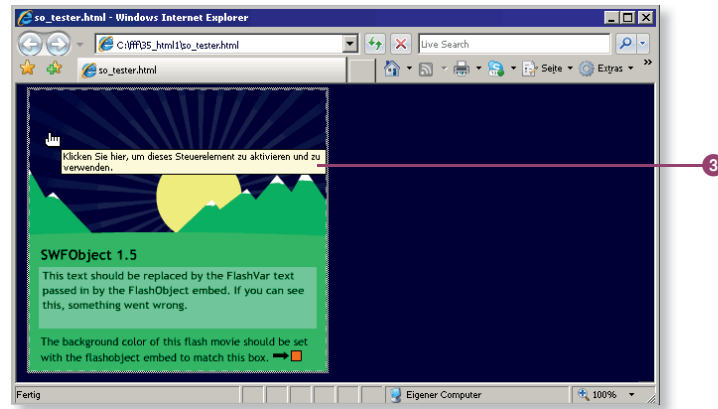
Schritt 4

Probleme beim Veröffentlichen mit Flash 8

Eolas-Patent

Das Eolas-Patent verbietet es dem Internet Explorer, interaktive Inhalte direkt in die Seite einzubetten. Andere Browser wie z. B. Mozilla Firefox oder Safari sind bisher nicht davon betroffen, was sich allerdings in Zukunft ändern könnte. Mit dem nächsten Service-Pack für Windows XP bzw. Windows Vista wird Microsoft im April 2008 diese Hürde wieder aus dem Internet Explorer entfernen. Welche Auswirkungen dies auf das SWFObject haben wird, können wir zu diesem Zeitpunkt nicht sagen.

Sollte der Internet Explorer nicht Ihr Standardbrowser sein, so öffnen Sie die Datei `so_tester.html` einmal testweise im Internet Explorer. Klicken Sie dazu mit der rechten Maustaste auf die Datei, wählen Sie dann im Kontextmenü **ÖFFNEN MIT** und aus der Auswahlliste den Internet Explorer.



▲ Abbildung 2

Im Internet Explorer muss der Flash-Film erst aktiviert werden, um abgespielt zu werden.

Schritt 5 SWFObject laden

Automatische Plug-in-Erkennung

Auch Flash CS3 bietet Ihnen eine automatische Plug-in-Erkennung. Diese beschränkt sich jedoch auf das Erkennen des Flash Players – ein automatisches Update wie beim SWFObject ist nicht integriert.

Ob Sie Flash 8 oder Flash CS3 benutzen – es gibt eine prinzipiell bessere und flexiblere Lösung, die Flash-Inhalte in eine HTML-Datei einzubetten: das SWFObject.

Hierbei handelt es sich um ein JavaScript, das die eben beschriebene Einbindung für Sie übernimmt. Zudem beinhaltet das SWFObject noch weitere Funktionen wie z. B. eine Plug-in-Erkennung, ob die richtige Version des Flash Players installiert ist. Wenn nötig, startet SWFObject sogar das automatische Update des Flash Players auf die neueste Version.

Für das weitere Vorgehen benötigen Sie die JavaScript-Datei des SWF-Objects. Wir haben sie Ihnen auf der Buch-DVD zur Verfügung gestellt. Kopieren Sie die Datei `SWFObject.js` in den von Ihnen angelegten Workshop-Ordner. Dieser sollte inzwischen neben der js-Datei eine html-, fla- und swf-Datei enthalten.

Schritt 6 HTML-Datei vorbereiten

Öffnen Sie `so_tester.html` in Ihrem Text- bzw. HTML-Editor, und löschen Sie alle überflüssigen Codezeilen, so dass lediglich das HTML-Gerüst übrigbleibt. Die Datei sollte nun wie folgt aussehen:

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

```

2: <head>
3: <meta http-equiv="Content-Type" content="text/html;
  charset=iso-8859-1" />
4: <title>so_tester.html</title>
5: </head>
6: <body bgcolor="#000033"></body>
7: </html>

```

Zur Einbindung des SWFObjects ergänzen Sie die benötigte JavaScript-Datei im `<head>`-Tag des HTML-Dokuments aus Schritt 6. Fügen Sie nach Zeile 4 die folgende Zeile ein:

```
<script type="text/javascript" src="swfobject.js"></script>
```

Anschließend fügen Sie die folgenden Zeilen innerhalb des `<body>`-Tags hinzu.

```

<div id="flashcontent"></div>
<script type="text/javascript">
// 
    var so = new SWFObject("so_tester.swf", "sotest", "300",
        "300", "8.0.24", "#FF6600");
    so.write("flashcontent");
// ]]&gt;
&lt;/script&gt;
</pre>
</div>
<div data-bbox="130 605 723 644" data-label="Text">
<p>Zunächst erzeugen Sie den <code>&lt;div&gt;</code>-Container <code>flashcontent</code>, in dem später der eigentliche Flash-Inhalt dargestellt wird.</p>
</div>
<div data-bbox="130 645 722 684" data-label="Text">
<p>Mit <code>new SWFObject</code> wird ein neues SWFObject erzeugt, das die Einbindung Ihrer <code>swf</code>-Datei in die HTML-Seite übernimmt.</p>
</div>
<div data-bbox="130 686 723 783" data-label="Text">
<p>Der erste Übergabeparameter steht für die <code>swf</code>-Datei, die angezeigt werden soll, der zweite für die ID des <code>&lt;embed&gt;</code>- bzw. <code>&lt;object&gt;</code>-Tags. Obwohl Sie die ID in diesem Workshop nicht weiter verwenden werden, müssen Sie an dieser Stelle eine eindeutige ID vergeben. Diese könnten Sie z. B. dazu benutzen, per JavaScript auf den eingebetteten Flash-Film zuzugreifen.</p>
</div>
<div data-bbox="130 784 722 843" data-label="Text">
<p>Parameter 3 und 4 beschreiben die Breite und Höhe des Flash-Films. Die zwei letzten übergebenen Werte legen die benötigte Version des Flash Players sowie die Hintergrundfarbe der <code>swf</code>-Datei fest.</p>
</div>
<div data-bbox="130 844 722 883" data-label="Text">
<p>Anschließend wird das erzeugte SWFObject per <code>so.write()</code> in den vorher angelegten Container geschrieben.</p>
</div>
<div data-bbox="744 94 800 139" data-label="Image">
<img alt="Icon of a CD-ROM."/>
</div>
<div data-bbox="801 92 946 155" data-label="Text">
<p>Zwischenergebnis:<br/>34_IN-HTML-EIN-<br/>BINDEN • SO_TES-<br/>TER_STEP06.HTML</p>
</div>
<div data-bbox="742 248 946 287" data-label="Section-Header">
<h2>Schritt 7<br/>SWFObject einbinden</h2>
</div>
<div data-bbox="744 312 800 357" data-label="Image">
<img alt="Icon of a CD-ROM."/>
</div>
<div data-bbox="801 309 952 373" data-label="Text">
<p>Zwischenergebnis:<br/>34_IN-HTML-EIN-<br/>BINDEN • SO_TESTER_<br/>STEP07.HTML</p>
</div>
<div data-bbox="752 401 849 417" data-label="Section-Header">
<h3>SWFObject</h3>
</div>
<div data-bbox="752 420 946 456" data-label="Text">
<p>Informationen zum SWF-Object erhalten Sie hier:</p>
</div>
<div data-bbox="752 458 973 552" data-label="List-Group">
<ul style="list-style-type: none;">
<li>► <a href="http://blog.deconcept.com/swfobject/">http://blog.deconcept.com/swfobject/</a></li>
<li>► <a href="http://blog.deconcept.com/swfobject/swfobject.zip">http://blog.deconcept.com/swfobject/swfobject.zip</a></li>
</ul>
</div>
<div data-bbox="752 619 835 636" data-label="Section-Header">
<h3>&lt;div&gt;-Tag</h3>
</div>
<div data-bbox="752 638 976 846" data-label="Text">
<p>Das <code>&lt;div&gt;</code>-Tag (<code>div</code> = division = Bereich) leitet in der HTML-Seite ein Block-Element ein. Dieses Element kann wiederum mehrere Block-Elemente beinhalten, die es zu einem Bereich gruppiert. Die Inhalte des <code>&lt;div&gt;</code>-Tags können Sie z. B. über den <code>align</code>-Parameter innerhalb des Bereichs ausrichten.</p>
</div>
<div data-bbox="678 936 956 954" data-label="Page-Footer">
<p>Flash-Filme in HTML einbinden 409</p>
</div>
```

Schritt 8

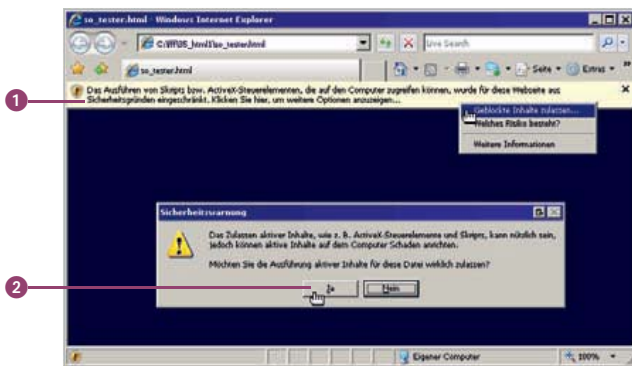
Das SWFObject testen

Transparenter Hintergrund

In manchen Fällen benötigen Sie einen transparenten Hintergrund. Wie Sie diesen erzeugen können, lernen Sie im nächsten Workshop.

Schauen Sie sich die neu erstellte HTML-Datei in einem Browser Ihrer Wahl an – die Webseite sollte in allen Browsern, die Sie ausprobieren, gleich aussehen.

Lediglich der Internet Explorer fordert Sie vor dem Anzeigen der Seite auf, die Ausführung von Skripten bzw. ActiveX-Steuerelementen freizugeben. Nach einem Rechtsklick auf die gelbe Leiste ❶ erscheint ein Kontextmenü. Wählen Sie hier GEBLOCKTE INHALTE ZULASSEN.... Danach müssen Sie noch im Popup-Fenster SICHERHEITSWARNUNG mit JA ❷ bestätigen.



▲ Abbildung 3
Sicherheitswarnung im Internet Explorer



▲ Abbildung 4
Die swf-Datei im Internet Explorer nach der Aktivierung der aktiven Inhalte

Dieses Phänomen tritt aber nur auf, wenn Sie die Datei lokal auf Ihrem Rechner ansehen. Sobald Sie die Datei auf Ihren Webserver hochgeladen und über die entsprechende URL aufgerufen haben, werden Sie diese Aufforderung nicht mehr sehen.

Zentriert, transparent und in High-Quality

Veröffentlichungsparameter beeinflussen

Es gibt immer wieder Anwendungsfälle, bei denen Sie die unterschiedlichen Parameter Ihres Flash-Films verändern müssen.

Sie können z. B. ganz ohne Probleme die Größe oder Skalierung des angezeigten SWF anpassen.

Dieser Workshop zeigt Ihnen, welche zusätzlichen Parameter Sie mit dem SWFObject beeinflussen können.

```
<script type="text/javascript" src="swfobject.js"></script>
</head>
<body bgcolor="#000033">
<div id="flashcontent"></div>
<script type="text/javascript">
// <![CDATA[
    var so = new SWFObject("so_tester.swf", "sotest",
        "100%",
        "100%", "8.0.24", "#FF6600");
    so.addParam("scale", "noscale");
    so.addParam("salign", "b");
    so.addParam("wmode", "transparent");
    so.addParam("quality", "high");
    so.write("flashcontent");
// ]]>
```

Zielsetzungen:

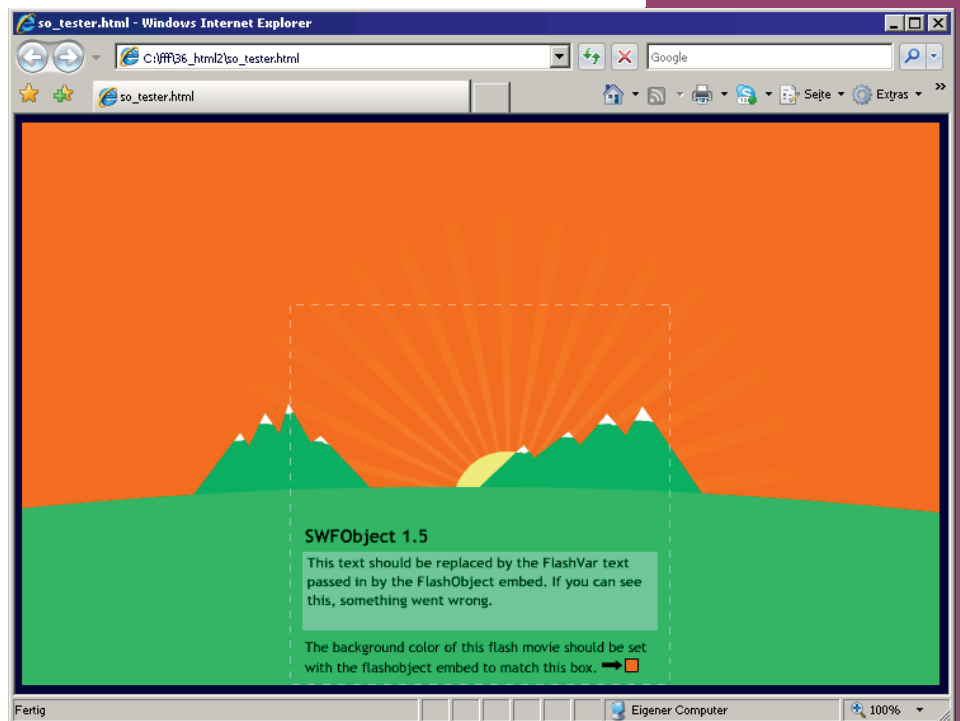
Größe des Flash-Films
ändern

Skalierung anpassen

SWF neu ausrichten

Qualität der Darstellung
ändern

[Ergebnis: 35_ZENTRIERT-
TRANSPARENT-HIGH-
QUALITY]



Schritt 1 Vorbereitungen

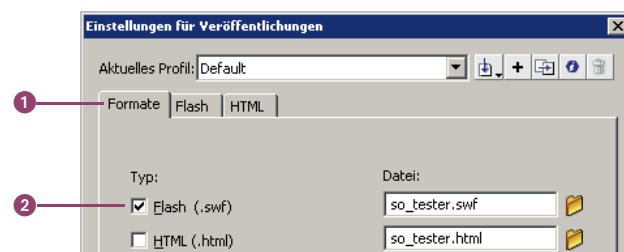
Im ersten Schritt legen Sie einen neuen Ordner auf Ihrer Festplatte an. Hier werden Sie die Beispiele aus diesem Workshop abspeichern. Nennen Sie den Ordner HTML2.

Als Beispieldatei kopieren Sie nun die Datei SO_TESTER.FLA von der Buch-DVD in den eben erstellten Ordner. Öffnen Sie dann die Datei in der Flash-Entwicklungsumgebung.

Schritt 2 Einstellungen für Veröffentlichungen

Jetzt vergewissern Sie sich, dass Flash beim Veröffentlichen des Flash-Films nur die zugehörige swf-Datei erzeugt. Öffnen Sie dazu über **[Strg]/[Alt]+[U]+[F12]** die EINSTELLUNGEN FÜR VERÖFFENTLICHUNGEN. Im Reiter FORMATE **1** darf ein Häkchen nur bei FLASH (.SWF) **2** gesetzt sein. Lassen Sie den Namen der Datei unverändert, und bestätigen Sie mit OK.

Abbildung 1 ►
Einstellungen für
Veröffentlichungen



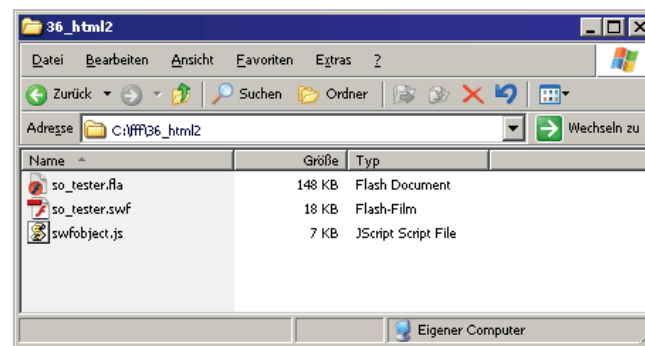
Schritt 3 Flash-Film veröffentlichen

Wenn Sie nun den Flash-Film mit **[U]+[F12]** veröffentlichen, erzeugt Flash die swf-Datei.

Schritt 4 SWFObject vorbereiten

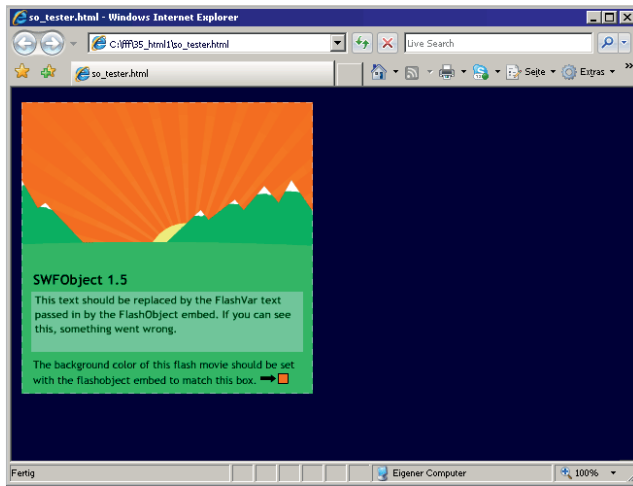
Kopieren Sie anschließend die zusätzlich benötigte Datei SWFObject.JS von der Buch-DVD in das am Anfang des Workshops erstellte Verzeichnis. Momentan sollten sich drei Dateien in diesem Ordner befinden: neben der soeben kopierten JavaScript-Datei auch die Dateien SO_TESTER.FLA und SO_TESTER.SWF.

Abbildung 2 ►
Bisher wurden drei Dateien
erstellt.



Damit Sie Ihre erstellte swf-Datei im Webbrowser testen können, fehlt nur noch die html-Datei. Wenn Sie den vorherigen Workshop bereits bearbeitet haben, kopieren Sie die dort erstellte Datei `SO_TESTER.HTML` in Ihren aktuellen Workshop-Ordner. Sollten Sie den Workshop noch nicht durchgeführt haben, können Sie auf die Beispieldatei von der Buch-DVD zurückgreifen.

Starten Sie `SO_TESTER.HTML` in Ihrem Standardbrowser. Sie sollten Folgendes sehen:



Schritt 5

Im Browser testen

◀ Abbildung 3

Die swf-Datei wird im Browser angezeigt.

Öffnen Sie die Datei `so_tester.html` jetzt in Ihrem Text- bzw. HTML-Editor. Die Datei sollte wie folgt aussehen:

```
1: <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="de"
   lang="de">
2: <head>
3: <meta http-equiv="Content-Type" content="text/html;
   charset=iso-8859-1" />
4: <title>so_tester.html</title>
5: <script type="text/javascript" src="swfobject.js"></script>
6: </head>
7: <body bgcolor="#000033">
8: <div id="flashcontent"></div>
9: <script type="text/javascript">
10: // 
11:     var so = new SWFObject("so_tester.swf", "sotest", "300",
        "300", "8.0.24", "#FF6600");
12:     so.write("flashcontent");
13: // ]]&gt;</pre>
</div>
<div data-bbox="742 546 833 564" data-label="Section-Header">
<h2>Schritt 6</h2>
</div>
<div data-bbox="742 566 922 584" data-label="Section-Header">
<h3>HTML-Datei öffnen</h3>
</div>
<div data-bbox="743 610 800 655" data-label="Image">
<img alt="Icon of a CD-ROM."/>
</div>
<div data-bbox="801 607 953 687" data-label="Text">
<p>Zwischenergebnis:<br/>35_ZENTRIERT-TRANSPARENT-HIGHQUALITY<br/>• <code>SO_TESTER_STEP06.HTML</code></p>
</div>
<div data-bbox="613 936 957 955" data-label="Page-Footer">
<p>Zentriert, transparent und in High-Quality <b>413</b></p>
</div>
```

```

14: </script>
15: </body>
16: </html>

```

Schritt 7 Größe des Flash-Films anpassen

Zwischenergebnis:
35_ZENTRIERT-TRANS-
PARENT-HIGHQUALITY
• SO_TESTER_STEP07.
HTML



Der bisherige Code sorgt dafür, dass das SWF in die html-Datei eingebettet und korrekt angezeigt wird. Nun werden Sie nachträglich einige Parameter verändern und sich das jeweilige Ergebnis ansehen.

Als Erstes ändern Sie in Zeile 11 der html-Datei die Größe des Flash-Films. Bislang haben Sie die Breite und Höhe des Films in Pixel angegeben. Tauschen Sie diese beiden Werte jeweils gegen den Wert 100% aus, so dass Zeile 11 nun wie folgt aussieht:

```

var so = new SWFObject("so_tester.swf", "sotest", "100%", "100%",
"8.0.24", "#FF6600");

```

Der Wert 100% für die Höhe und Breite sorgt dafür, dass der Film über die ganze Fläche des Browsers skaliert wird.

Speichern Sie die html-Datei ab.

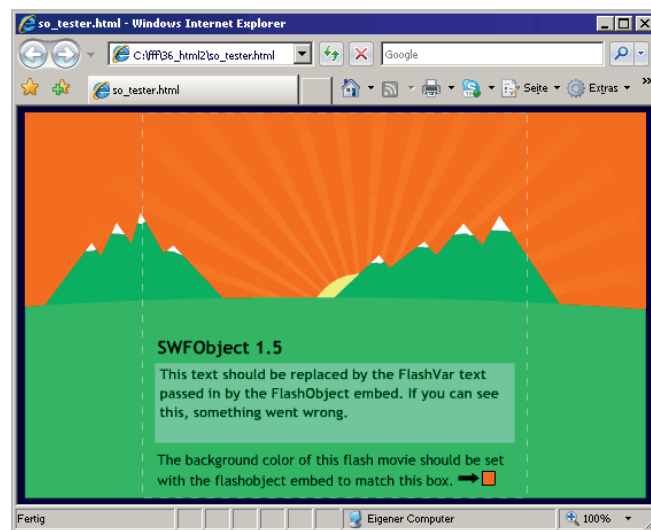


Abbildung 4 ►

Der Flash-Film wird auf die Größe des Browserfensters skaliert. Die Grafiken und Texte werden somit größer dargestellt, als in der fla-Datei angelegt.

Wenn Sie nun die Größe des Browserfensters verändern, skaliert der Flash-Film mit, so dass er immer das komplette Browserfenster ausfüllt.

Schritt 8 Skalierung abstellen

Dass der Flash-Film bzw. seine Inhalte skaliert werden, ist in den meisten Fällen nicht erwünscht. Sie unterbinden die Skalierung, indem Sie den

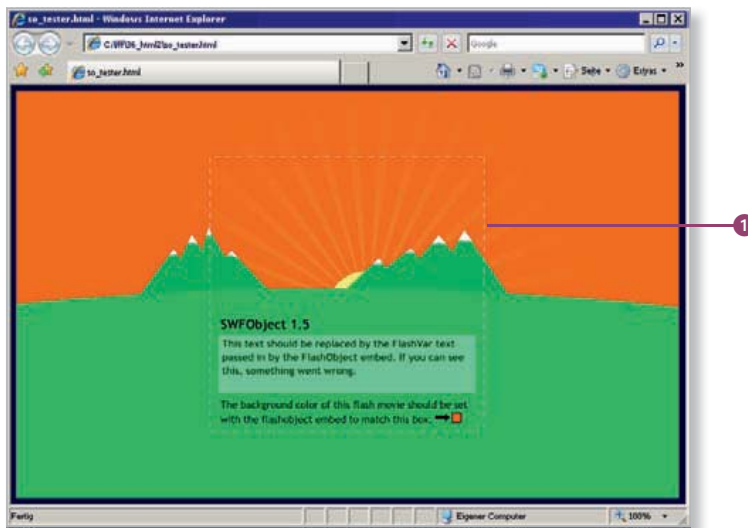
scale-Parameter auf `noscale` setzen. Fügen Sie hierzu unter Zeile 11 die folgende Zeile neu ein:

```
so.addParam("scale", "noscale");
```

Betrachten Sie das Ergebnis im Browser. Der Flash-Film sollte nun nicht mehr skaliert, sondern ab sofort zentriert im Browser angezeigt werden.

Sollte wie in diesem Fall der Container für den Flash-Film größer sein als die eigentliche swf-Datei, so wird auch alles angezeigt, was normalerweise außerhalb der Bühne liegt. Der gestrichelte Rahmen ❶ zeigt die Grenze der swf-Datei an.

Speichern Sie die vorgenommenen Änderungen ab.



Zwischenergebnis:
35_ZENTRIERT-TRANSPARENT-HIGHQUALITY
• SO_TESTER_STEP08.
HTML

◀ Abbildung 5

Das SWF wird immer noch im kompletten Browserfenster angezeigt – doch jetzt nicht mehr skaliert, sondern in der Originalgröße.

Bei der eben gezeigten Darstellung wird der Flash-Film standardmäßig zentriert im Browser angezeigt. Um eine andere Ausrichtung zu erzwingen, verwenden Sie den `align`-Parameter. Die möglichen Werte für diesen Parameter sind:

- ▶ T = Top (oben ausgerichtet)
- ▶ B = Bottom (unten ausgerichtet)
- ▶ L = Left (links ausgerichtet)
- ▶ R = Right (rechts ausgerichtet)

– sowie Kombinationen der genannten Werte, z. B.:

- ▶ TL = Top-Left (oben links ausgerichtet)
- ▶ BR = Bottom-Right (unten rechts ausgerichtet)

Schritt 9

Ausrichtung am unteren Browserrand



Zwischenergebnis:
35_ZENTRIERT-TRANSPARENT-HIGHQUALITY
• SO_TESTER_STEP09.
HTML



Abbildung 6 ▲

Der Flash-Film wird am unteren Rand ausgerichtet.

Schritt 10

Flash-Film transparent darstellen

Zwischenergebnis:
35_ZENTRIERT-TRANSPARENT-HIGHQUALITY
• SO_TESTER_STEP10.
HTML



Wmode = transparent

Der `wmode`-Parameter kann bei einigen älteren Browsern zu Problemen führen. Nähere Informationen dazu finden Sie hier:

http://kb.adobe.com/selfservice/viewContent.do?externalId=tn_14201&sliceId=2 (englisch)

Abbildung 7 ►

Jetzt ist der Hintergrund des SWF transparent, und nicht mehr orange – dahinterliegende HTML-Elemente würden durchscheinen.

Sie werden nun im nächsten Schritt den Flash-Film am unteren Rand des Browsers ausrichten. Dazu fügen Sie – am besten direkt unter der im vorherigen Schritt erzeugten Zeile – eine weitere Zeile in die html-Datei ein:

```
so.addParam("salign", "B");
```

Speichern Sie die html-Datei ab, und öffnen Sie sie anschließend im Browser. Der Flash-Film sollte am unteren Rand des Browsers »kleben«.

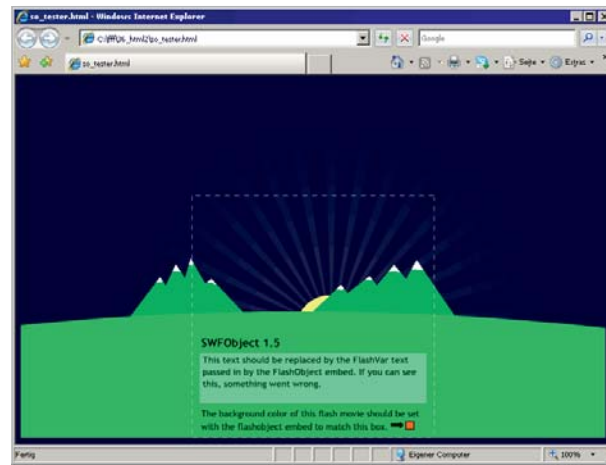
In manchen Fällen benötigen Sie einen transparenten Hintergrund, so dass hinter dem Flash-Film liegende HTML-Elemente noch zu sehen sind. Ein gutes Beispiel für diesen Anwendungsfall sind Banner auf einer Webseite. Diese liegen oftmals über dem eigentlichen Inhalt, sollen aber gleichzeitig nicht die komplette Seite verdecken, sondern die darunterliegenden Elemente durchscheinen lassen.

Diesen Effekt erreichen Sie, indem Sie den `wmode`-Parameter (windowless mode) auf `transparent` setzen. Fügen Sie hierzu Ihrer html-Datei eine weitere Zeile hinzu:

```
so.addParam("wmode", "transparent");
```

Speichern Sie die Änderungen unter dem Namen `so_tester.html` im Workshop-Ordner ab, und öffnen Sie die Datei danach im Browser.

Der Hintergrund des Flash-Films sollte nun in derselben Farbe, die im `<body>`-Tag der hmtl-Datei angegeben ist, erscheinen, in diesem Fall Dunkelblau.



Flash-Filme können mit vier Qualitätseinstellungen abgespielt werden:

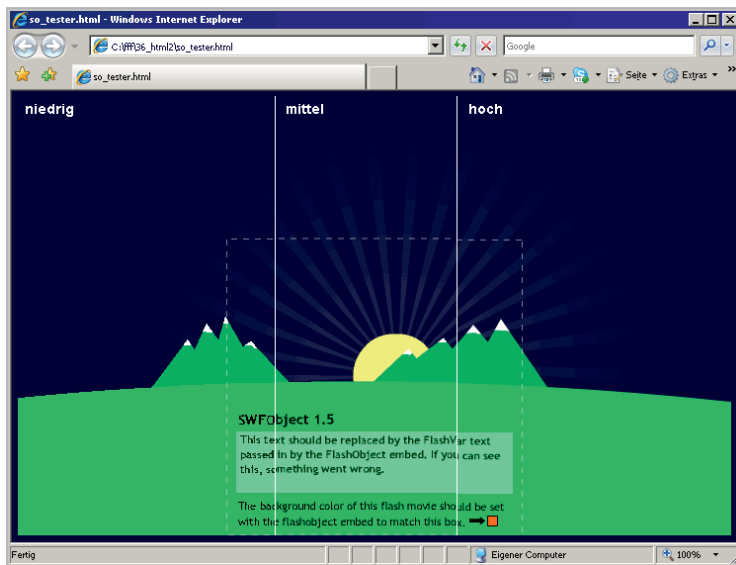
- ▶ low (niedrig)
- ▶ medium (mittel)
- ▶ high (hoch)
- ▶ best (beste)

Dieser Parameter gibt an, mit welcher Qualität der Flash Player das SWF rendert. Wenn Sie die Qualität z. B. auf `low` setzen, werden u. a. die eingebetteten Bitmaps nicht weichgezeichnet. Diese und einige andere Veränderungen in der Darstellung lassen den Flash-Film wesentlich »flüssiger« ablaufen. Wollten Sie ein Spiel in Flash programmieren, das auch auf älteren Computern ohne Ruckeln abläuft, so würden Sie die Qualität auf `low` bzw. `medium` stellen.

In diesem Beispiel müssen Sie aber nicht auf die Performance achten und stellen die Qualität deshalb auf `high`. Hierzu fügen Sie eine weitere Zeile unter den zuletzt eingefügten Code in die `html`-Datei ein:

```
so.addParam("quality", "high");
```

Spiele Sie ein bisschen mit dieser Einstellung, damit Sie wissen, welche Auswirkungen die Änderung der Qualität hat.



Schritt 11

Qualität beeinflussen



Zwischenergebnis:
35_ZENTRIERT-TRANSPARENT-HIGHQUALITY
• SO_TESTER_STEP11.
HTML

◀ Abbildung 8

Unterschiedliche Qualitätsstufen

Flash Player-Versionserkennung

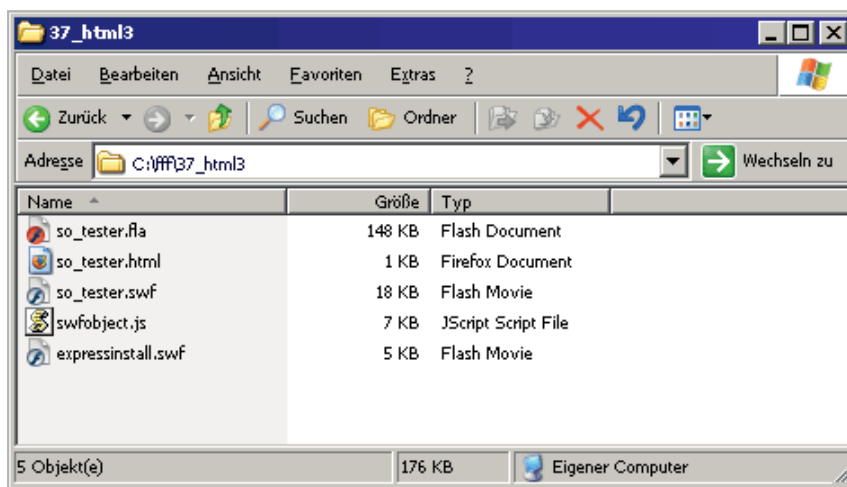
Automatische Erkennung und Installation des Flash Players

```
<script type="text/javascript" src="swfobject.js"></script>
</head>
<body bgcolor="#000033">
<div id="flashcontent"><font color="#ffffff">Sie haben entweder
kein Flash installiert oder Ihre Version des Flash Players ist
nicht ausreichend - bitte aktualisieren Sie Ihr Flash-Plug-in.
</font></div>
<script type="text/javascript">
// 
    var so = new SWFObject("so_tester.swf", "sotest", "100%",
    "100%", "8.0.24", "#FF6600");
    so.addParam("scale", "noscale");
    so.addParam("salig", "b");
    so.addParam("wmode", "transparent");
    so.addParam("quality", "high");
    so.write("flashcontent");</pre>
</div>
<div data-bbox="664 246 867 605" data-label="Text">
<p><i>Ein weiterer wichtiger Punkt, wenn Sie Ihre Flash-Seite im Internet veröffentlichen wollen, ist der Plug-in-Check. Dieser stellt sicher, dass der Besucher Ihrer Seite immer den Flash Player in der richtigen Version installiert hat und so die Inhalte Ihrer Seite korrekt angezeigt werden. In diesem Workshop lernen Sie, wie einfach diese Überprüfung mit dem SWF-Object zu implementieren ist und wie Sie, wenn nötig, die Flash-Player-Version aktualisieren.</i></p>
</div>
<div data-bbox="49 576 639 878" data-label="Image">
<img alt="Screenshot of a Windows Internet Explorer browser window showing an Adobe Flash Player update dialog box. The dialog box asks 'Möchten Sie Adobe Flash Player 9 installieren?' (Do you want to install Adobe Flash Player 9?) with 'Ja' (Yes) and 'Nein' (No) buttons. The browser window title is 'so_tester.html - Windows Internet Explorer' and the address bar shows 'C:\VFP37_html3\so_tester.html'."/>
</div>
<div data-bbox="664 667 778 683" data-label="Section-Header">
<h3>Zielsetzungen:</h3>
</div>
<div data-bbox="674 686 859 786" data-label="List-Group">
<ul>
<li>Versionscheck des Flash Players</li>
<li>Update des Flash Players</li>
<li>Landingpage nach erfolgreicher Installation</li>
</ul>
</div>
<div data-bbox="664 800 862 834" data-label="Text">
<p>[Ergebnis: 36_FLASH-PLAYER-VERSIONSERKENNUNG]</p>
</div>
```

Im ersten Schritt legen Sie einen neuen Ordner auf Ihrer Festplatte an. Hier werden Sie die Beispiele aus diesem Workshop abspeichern. Nennen Sie den Ordner HTML3.

Auf der Buch-DVD finden Sie den dazugehörigen Ordner. Kopieren Sie von dort die folgenden Dateien in Ihren soeben erstellten Workshop-Ordner:

- ▶ SO_TESTER.FLA
- ▶ SO_TESTER.HTML
- ▶ SO_TESTER.SWF
- ▶ SWFOBJECT.JS
- ▶ EXPRESSINSTALL.SWF



Schritt 1

Vorbereitungen

◀ **Abbildung 1**

Für diesen Workshop werden fünf Dateien benötigt.

Öffnen Sie die Datei SO_TESTER.HTML in Ihrem Text- bzw. HTML-Editor. Bisher beinhaltet diese Datei folgenden Code:

```
1: <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="de"
   lang="de">
2: <head>
3: <meta http-equiv="Content-Type" content="text/html;
   charset=iso-8859-1" />
4: <title>so_tester.html</title>
5: <script type="text/javascript" src="swfobject.js"></
   script>
6: </head>
7: <body bgcolor="#000033">
8: <div id="flashcontent"></div>
9: <script type="text/javascript">
```

Schritt 2

HTML-Datei öffnen

Flash-Player-Version

Um die bei Ihnen aktuell installierte Version des Flash Players herauszufinden, besuchen Sie folgende Webseite:

<http://www.adobe.com/products/flash/about/>

Hier erhalten Sie auch Informationen über die neuesten Versionen des Flash-Plug-ins für die verschiedenen Betriebssysteme.

```
10: // 
11:   var so = new SWFObject("so_tester.swf", "sotest", "100%",
12:     "100%", "8.0.24", "#FF6600");
13:   so.addParam("scale", "noscale");
14:   so.addParam("salign", "B");
15:   so.addParam("wmode", "transparent");
16:   so.addParam("quality", "high");
17:   so.write("flashcontent");
18: // ]&gt;
19: &lt;/script&gt;
20: &lt;/body&gt;
21: &lt;/html&gt;</pre></div><div data-bbox="270 328 867 385" data-label="Text"><p>In Zeile 11 wird als fünfter Parameter 8.0.24 mit übergeben. Das heißt, dass der Flash-Film mindestens den Flash Player in der Version 8.0.24 voraussetzt.</p></div><div data-bbox="88 427 246 465" data-label="Section-Header"><h2>Schritt 3<br/>Im Browser testen</h2></div><div data-bbox="270 428 867 466" data-label="Text"><p>Stellen Sie sicher, dass Ihr Webbrowser über die im vorherigen Schritt angesprochene Plug-in-Version (oder eine neuere) verfügt.</p></div><div data-bbox="270 468 867 524" data-label="Text"><p>Nachdem Sie sichergestellt haben, dass Sie über eine ausreichend hohe Version des Flash Players verfügen, starten Sie die Datei SO_TESTER.HTML in Ihrem Standardbrowser.</p></div><div data-bbox="270 539 838 881" data-label="Image"><img alt="Screenshot of a Windows Internet Explorer browser window displaying the so_tester.html file. The browser window shows a dark blue background with a green mountain range and a yellow sun. A dashed white box is centered on the screen, containing the text 'SWFObject 1.5' and two lines of instructions: 'This text should be replaced by the FlashVar text passed in by the FlashObject embed. If you can see this, something went wrong.' and 'The background color of this flash movie should be set with the flashobject embed to match this box.' followed by a small orange square. The browser's address bar shows the file path 'C:\FFF36_html2\so_tester.html' and the title bar says 'so_tester.html - Windows Internet Explorer'."/></div><div data-bbox="142 802 246 818" data-label="Caption"><p>Abbildung 2 ►</p></div><div data-bbox="65 818 247 883" data-label="Text"><p>Die swf-Datei im Browserfenster. Sie wird zentriert und am unteren Rand des Bildschirms angezeigt.</p></div><div data-bbox="47 937 211 954" data-label="Page-Footer"><p>420 Flash im Internet</p></div>
```

SWFObject-Parameter

Für den Fall, dass Sie den vorherigen Workshop noch nicht bearbeitet haben, hier eine kurze Erklärung der bisher verwendeten SWFObject-Parameter:

100% Breite und Höhe bewirken, dass das SWF das gesamte Browserfenster ausfüllt.

`scale = noscale` bewirkt, dass die Inhalte des SWF nicht skaliert werden, sondern der Bereich um die Bühne herum mit angezeigt wird.

`align = b` richtet den Flash-Film am unteren Rand des Browserfensters aus.

Durch `wmode = transparent` bleibt der sonst orange Hintergrund des Flash-Films transparent.

Mit `quality = high` wurde die Qualität des Flash-Films auf »hoch« gesetzt.

Die Überprüfung, ob die richtige Version des Flash Players installiert ist, übernimmt das SWFObject automatisch. Sie müssen lediglich noch festlegen, was passieren soll, wenn die Prüfung ergibt, dass der Besucher mit einer veralteten Flash Player-Version Ihre Seite öffnet. Zu diesem Zweck ändern Sie die Zeile 8 der html-Datei folgendermaßen ab:

```
<div id="flashcontent"><font color="#ffffff">Sie haben entweder  
kein Flash installiert oder Ihre Version des Flash Players ist  
nicht ausreichend - bitte aktualisieren Sie Ihr Flash-Plug-in.</  
font></div>
```

Der Text wird nur angezeigt, wenn der Besucher ein nicht ausreichendes bzw. gar kein Flash-Plug-in installiert hat.

Wenn Sie nun die html-Datei erneut im Browser öffnen, werden Sie keinen Unterschied zu vorher erkennen. Dies liegt daran, dass Sie eine ausreichende Flash-Player-Version installiert haben.

Ältere Flash-Plug-ins installieren

Um Tests mit unterschiedlichen Versionen des Flash Players durchzuführen, bietet es sich an, einen sogenannten Plug-in-Switcher zu installieren. Damit sind Sie in der Lage, auch noch nachträglich Ihre Flash-Player-Version zu ändern: Sie können dann zwischen allen installierten Versionen wählen.

<http://www.pluginswitcher.de>

Ältere Versionen des Flash Players zum Download:

http://kb.adobe.com/selfservice/viewContent.do?externalId=tn_14266&sliceId=2

Schritt 4 Alternativen Inhalt festlegen

Schritt 5 Im Browser testen

Schließen Sie den Browser wieder, installieren Sie Version 7 des Flash-Plug-ins, und öffnen Sie die Seite erneut. Der Text, den Sie in Schritt 4 als alternativen Inhalt angegeben haben, sollte nun zu sehen sein **1**.



Abbildung 3 ►

Diese Meldung wird angezeigt, wenn im Browser nicht das richtige Flash-Plug-in installiert ist.

Schritt 6 **Flash-Plug-in automatisch aktualisieren**

Wie am Anfang des Workshops erwähnt, können Sie das SWFObject auch dazu benutzen, von einer veralteten auf eine neue Version des Flash Players zu aktualisieren. Hier kommt die in Schritt 1 kopierte Datei EXPRESSINSTALL.SWF mit ins Spiel.

Öffnen Sie die Datei SO_TESTER.HTML, und fügen Sie folgenden Code zwischen Zeile 15 und 16 ein:

```
so.useExpressInstall("expressinstall.swf");
```

In dieser Zeile wird die Datei EXPRESSINSTALL.SWF eingebunden, die sich um die automatische Aktualisierung des Flash-Plug-ins kümmert.

Speichern Sie die vorgenommenen Änderungen in der html-Datei. Öffnen Sie die Datei im Browser. Sie sollten jetzt gefragt werden, ob Sie Ihre Flash-Player-Version aktualisieren möchten. Bestätigen Sie im Pop-up-Fenster mit JA **2**.

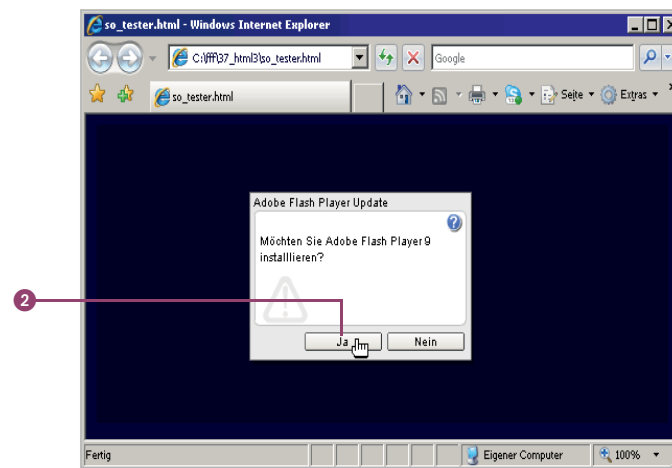
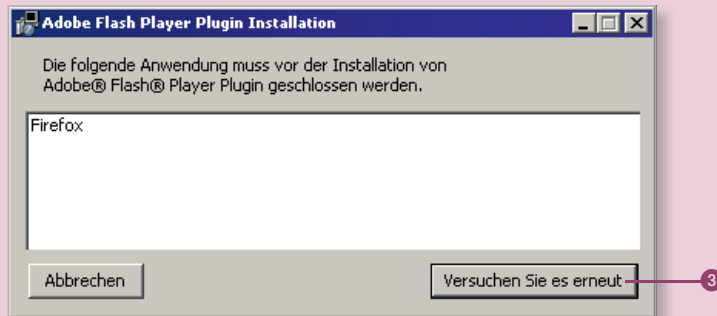


Abbildung 4 ►

Sie werden gefragt, ob das Flash-Plug-in aktualisiert werden soll.

Firefox

Sollten Sie die Seite mit dem Firefox-Browser geöffnet haben, werden Sie nun aufgefordert, den Browser zu schließen, bevor die Installation durchgeführt werden kann. Schließen Sie also den Browser, und bestätigen Sie im Popup-Fenster mit **VERSUCHEN SIE ES ERNEUT** ³. Nach erfolgreicher Installation des Plug-ins werden Sie zurück auf die vorher besuchte Seite geführt.



▲ Abbildung 5

Verwenden Sie Firefox, muss der Browser geschlossen werden. Über **VERSUCHEN SIE ES ERNEUT** wird anschließend der Flash-Film angezeigt.

In manchen Fällen wollen Sie den Besucher vielleicht noch »abfangen«, bevor er nach der Installation wieder auf die vorherige Seite geführt wird, um ihm zu erklären, was soeben passiert ist.

Zu diesem Zweck öffnen Sie ein weiteres Mal die Datei `so_TESTER.HTML` in Ihrem Editor und fügen unterhalb der Anweisung `so.useExpressInstall` ("`expressinstall.swf`") folgende Zeile hinzu:

```
so.setAttribute("xiRedirectUrl", "http://www.flashfastforward.de/  
swfobject/landingpage.html");
```

In dieser Zeile nutzen Sie einen neuen Funktionsaufruf des `SWFObjects`: `setAttribute()`. In diesem Fall setzen Sie darüber das `SWFObject`-Attribut `xiRedirectUrl`. Dieses erwartet als Wert eine absolute URL zu der Datei, die Sie nach der erfolgreichen Installation anzeigen möchten – die sogenannte Landingpage.

Speichern Sie die Änderungen an der html-Datei ab.

Die noch benötigte Landingpage werden Sie in diesem Schritt erzeugen. Öffnen Sie dafür Ihren HTML- oder Text-Editor, und legen Sie eine leere

Schritt 7

Umleitung nach erfolgreicher Installation

Schritt 8

Erstellen der Landingpage

Landingpage anpassen

Das Codebeispiel zeigt eine URL, die auf die Webseite des Buches verlinkt. Wenn Sie diesen Wert ändern und stattdessen auf Ihre eigene Domain weiterleiten wollen, müssen Sie an dieser Stelle Ihren Domainnamen einsetzen.

html-Datei an. Innerhalb des `<body>`-Tags fügen Sie einen kurzen Text hinzu, so dass die fertige Datei wie folgt aussieht:

```
1: <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="de"
   lang="de">
2: <head>
3: <meta http-equiv="Content-Type" content="text/html;
   charset=iso-8859-1" />
4:   <title>landingpage / aktualisierung erfolgreich abge-
   schlossen</title>
5: </head>
6: <body bgcolor="#000033">
7: <font color="#ffffff">Glückwunsch! Soeben wurde bei Ihnen die
   neuste Flash Player Version installiert.<br>
8: Hier geht es weiter zur Seite: <a href="so_tester.html">
   SWFObject Test</a></font>
9: </body>
10: </html>
```

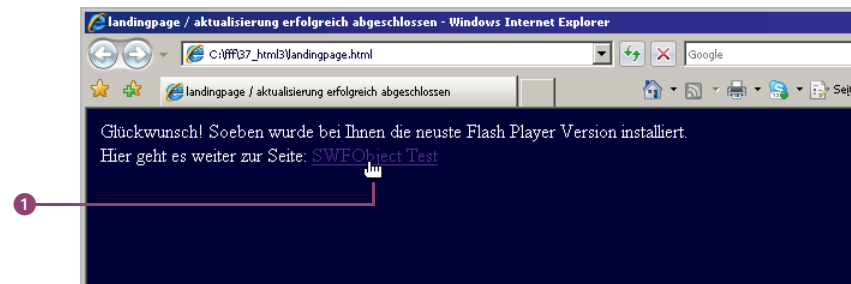
Speichern Sie diese Datei als `LANDINGPAGE.HTML` im Workshop-Ordner.

Um dieses Feature zu testen, müssen Sie entweder die eben erstellte Datei auf Ihren Webserver hochladen oder Sie benutzen bei der Umleitung die URL, die auf die Buch-Seite verweist.

In beiden Fällen sollten Sie beim Öffnen von `SO_TESTER.HTML` nach der erfolgreichen Installation des Flash-Plug-ins die eben erstellte Seite sehen. Der Link ① führt den Besucher dann wieder auf die eigentliche Seite.

Abbildung 6 ►

Erst über einen Link gelangen die User nun zu Ihrer swf-Datei.



Kommunikation mit HTML

Variablen zwischen HTML und Flash übergeben

Es passiert immer wieder, dass Sie einen oder mehrere Werte von außen an den Flash-Film übermitteln müssen – z. B., wenn Sie sogenannte »Deep-links« auf Ihrer Seite umsetzen möchten. In diesem Workshop lernen Sie, wie Sie mit Hilfe des SWFObjects ganz einfach Variablen aus HTML heraus an den Flash-Film weitergeben.

```
<script type="text/javascript" src="swfobject.js"></script>
</head>
<body bgcolor="#000033">
<div id="flashcontent"><font color="#ffffff">Sie haben entweder
kein Flash installiert oder Ihre Version des Flash Players ist
nicht ausreichend - bitte aktualisieren Sie Ihr Flash-Plug-in.
</font></div>
<script type="text/javascript">
// 
    var so = new SWFObject("so_tester.swf", "sotest", "100%",
    "100%", "8.0.24", "#FF6600");
    so.addParam("scale", "noscale");
    so.addParam("salign", "b");</pre>
</div>
<div data-bbox="214 554 324 570" data-label="Section-Header">
<h3>Zielsetzungen:</h3>
</div>
<div data-bbox="140 573 324 692" data-label="List-Group">
<ul>
<li>Variable an den Flash-Film übergeben</li>
<li>Variable im Flash-Film entgegennehmen</li>
<li>Anzeigen der Variable im Flash-Film</li>
</ul>
</div>
<div data-bbox="157 706 324 740" data-label="Text">
<p>[Ergebnis: 37_KOMMUNIKATION]</p>
</div>
<div data-bbox="347 519 951 878" data-label="Image">
<img alt="Screenshot of a Windows Internet Explorer browser window displaying a Flash movie. The browser title is 'so_tester.html - Windows Internet Explorer'. The address bar shows 'C:\VFF38_html4\so_tester.html'. The Flash movie has a dark blue background with green mountains and a yellow sun. A dashed white box highlights a text area within the movie. The text area contains the text 'SWFObject 1.5' and 'Hier steht dann Ihr Text!'. Below the text area, there is a small orange square and the text 'The background color of this flash movie should be set with the flashobject embed to match this box.' The browser status bar at the bottom shows 'Fertig', 'Eigener Computer', and '100%'."/>
</div>
```

Schritt 1 Vorbereitungen

Im ersten Schritt legen Sie einen neuen Ordner auf Ihrer Festplatte an. Hier werden Sie die von Ihnen erstellten Beispiele zu diesem Workshop speichern.

Wenn Sie die vorherigen Workshops zum Thema SWFObject bereits bearbeitet haben, können Sie einfach auf die dort benutzten Dateien zurückgreifen. Andernfalls finden Sie auf der Buch-DVD den zu diesem Workshop gehörenden Ordner. Kopieren Sie von dort die folgenden Dateien in Ihren soeben erstellten Workshop-Ordner:

- ▶ SO_TESTER.FLA
- ▶ SO_TESTER.HTML
- ▶ SO_TESTER.SWF
- ▶ SWFOBJECT.JS
- ▶ EXPRESSINSTALL.SWF

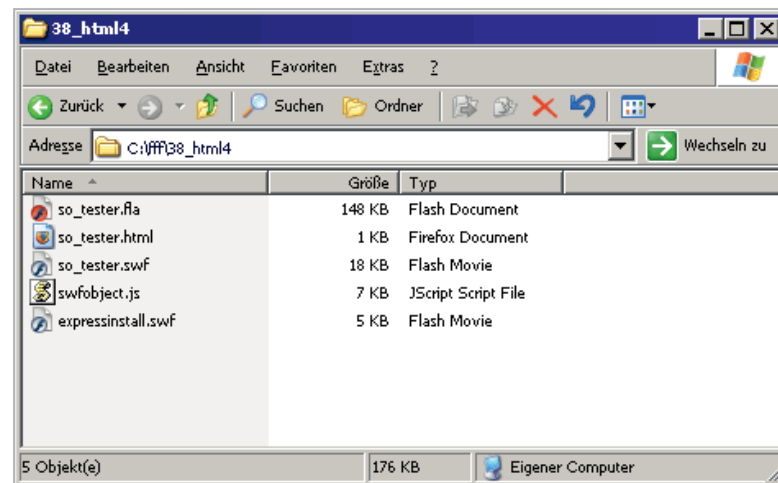


Abbildung 1 ▶
Alle Dateien dieses
Workshops

Schritt 2 HTML-Datei öffnen

Öffnen Sie die Datei SO_TESTER.HTML in Ihrem Text- bzw. HTML-Editor. Die Datei sollte wie folgt aussehen:

Zwischenergebnis:
37_KOMMUNIKATION
• SO_TESTER_STEP02.
HTML



```
1: <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="de"
   lang="de">
2: <head>
3: <meta http-equiv="Content-Type" content="text/html;
   charset=iso-8859-1" />
4:   <title>so_tester.html</title>
5:   <script type="text/javascript" src="swfobject.js">
   </script>
6: </head>
```

```

7: <body bgcolor="#000033">
8: <div id="flashcontent"><font color="#ffffff">Sie haben ent-
  weder kein Flash installiert oder Ihre Version des Flash
  Players ist nicht ausreichend - bitte aktualisieren Sie Ihr
  Flash-Plug-in.</font></div>
9: <script type="text/javascript">
10: // 
11:   var so = new SWFObject("so_tester.swf", "sotest", "100%",
     "100%", "8.0.24", "#FF6600");
12:   so.addParam("scale", "noscale");
13:   so.addParam("salign", "B");
14:   so.addParam("wmode", "transparent");
15:   so.useExpressInstall( "expressinstall.swf");
16:   so.write("flashcontent");
17: // ]]&gt;
18: &lt;/script&gt;
19: &lt;/body&gt;
20: &lt;/html&gt;
</pre>
</div>
<div data-bbox="129 467 722 506" data-label="Text">
<p>Wenn Sie die HTML-Datei im Browser Ihrer Wahl testen, sollten Sie Folgendes sehen:</p>
</div>
<div data-bbox="742 466 913 504" data-label="Section-Header">
<h3>Schritt 3</h3>
<h4>Im Browser testen</h4>
</div>
<div data-bbox="132 519 600 782" data-label="Image">
<img alt="Screenshot of a Windows Internet Explorer browser window displaying a Flash movie. The movie shows a stylized landscape with green hills, a yellow sun, and a dark blue sky. A dashed white box is overlaid on the movie, containing text that reads: 'SWFObject 1.5. This text should be replaced by the FlashVar text passed in by the FlashObject embed. If you can see this, something went wrong. The background color of this flash movie should be set with the flashobject embed to match this box'. A red circle with the number '1' points to the text inside the box. The browser window title is 'so_tester.html - Windows Internet Explorer'."/>
</div>
<div data-bbox="742 734 870 784" data-label="Caption">
<p>◀ <b>Abbildung 2</b><br/>Die swf-Datei im<br/>Internet Explorer</p>
</div>
<div data-bbox="129 804 722 884" data-label="Text">
<p>Achten Sie auf den Text ① innerhalb der hellgrünen Box. Momentan wird dieser noch statisch innerhalb der Flash-Datei gesetzt. In den nächsten Schritten dieses Workshops werden Sie eine Variable aus HTML heraus in Flash setzen und innerhalb des Textfeldes anzeigen.</p>
</div>
<div data-bbox="714 936 957 954" data-label="Page-Footer">
<p>Kommunikation mit HTML <b>427</b></p>
</div>
```

SWFObject-Parameter

Für den Fall, dass Sie den vorherigen Workshop noch nicht bearbeitet haben, hier eine kurze Erklärung der bisher verwendeten SWFObject-Parameter:

100% Breite und Höhe bewirken, dass das SWF das gesamte Browserfenster ausfüllt.

`scale = noscale` bewirkt, dass die Inhalte des SWF nicht skaliert werden, sondern der Bereich um die Bühne herum mit angezeigt wird.

`salign = b` richtet den Flash-Film am unteren Rand des Browserfensters aus.

Durch `wmode = transparent` bleibt der sonst orange Hintergrund des Flash-Films transparent.

Mit `quality = high` wurde die Qualität des Flash-Films auf »hoch« gesetzt.

Schritt 4 Textfeld benennen

Achtung, nicht veröffentlichen

In diesem Fall ist es wichtig, dass Sie den Film nicht veröffentlichen, sondern lediglich testen. Andernfalls würden Sie Ihre abgeänderte HTML-Datei überschreiben.

Öffnen Sie in der Flash-Entwicklungsumgebung die in Ihrem Workshop-Ordner befindliche Datei `so_tester.fla`.

Wählen Sie nun das eben angesprochene Textfeld ❶ aus. Stellen Sie sicher, dass DYNAMISCHER TEXT ❷ eingestellt ist und das Feld den Namen »flashVar_txt« ❸ trägt.

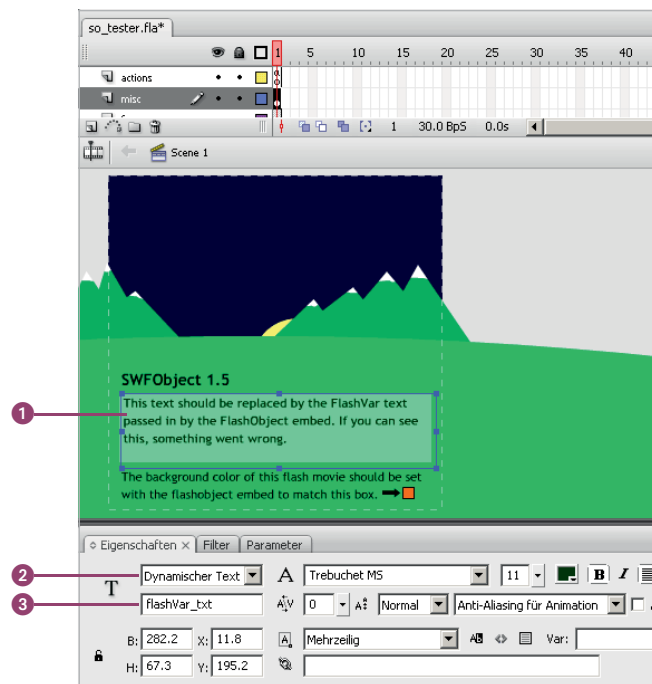


Abbildung 3 ▶
Dynamisches Textfeld

Testen Sie anschließend den Flash-Film über `Strg`/`⌘` + `↵`. Nach erfolgreicher Kompilierung öffnet sich das Flash-Player-Fenster. Schließen Sie dieses gleich wieder.

Später soll die aus HTML an den Flash-Film übergebene Variable in Flash weiterverarbeitet und dem Textfeld aus dem vorherigen Schritt zugewiesen werden. Wählen Sie daher das erste Bild der »actions«-Ebene auf der Zeitleiste aus und öffnen über FENSTER • AKTIONEN das AKTIONEN-FENSTER. Hier sollten Sie nachfolgendes Skript vorfinden:

```
3:  stop();
4:
5:  if (flashVarText != undefined)
6:  {
7:      flashVar_txt.text = flashVarText;
8:  }
```

Das `stop()` in Zeile 3 lässt den Film anhalten. In Zeile 5 wird mit einer `if`-Anweisung überprüft, ob die Variable `flashVarText` ungleich `undefined` ist. D.h., es wird kontrolliert, ob die Variable initialisiert wurde, ob ihr also ein Wert zugewiesen wurde. Wenn das der Fall ist – die `if`-Klausel also `true` zurückliefert – wird der `text` des Textfeldes `flashVar_txt` mit dem Inhalt dieser Variable gefüllt

Jetzt müssen Sie den Flash-Film ein weiteres Mal über `Strg`/`⌘` + `↵` testen. Das sich öffnende Fenster schließen Sie gleich wieder.

Jetzt fehlt Ihnen noch die eigentliche Übergabe der Variable und ihres Wertes an den Flash-Film. Hierzu öffnen Sie erneut die Datei `so_tester.html` in Ihrem Editor und fügen unterhalb der Zeile 15 folgenden Code hinzu:

```
so.addVariable("flashVarText", "Hier steht dann Ihr Text!");
```

Speichern Sie die geänderte `html`-Datei ab.

Öffnen Sie die soeben gespeicherte `html`-Datei in einem Webbrowser Ihrer Wahl. Der in Schritt 6 an den Flash-Film übergebene Variablenwert »Hier steht dann Ihr Text!« sollte in dem hellgrün unterlegten Textfeld ④ zu sehen sein.

Schritt 5

Variable in Flash entgegennehmen

Schritt 6

Variable an den Flash-Film übergeben

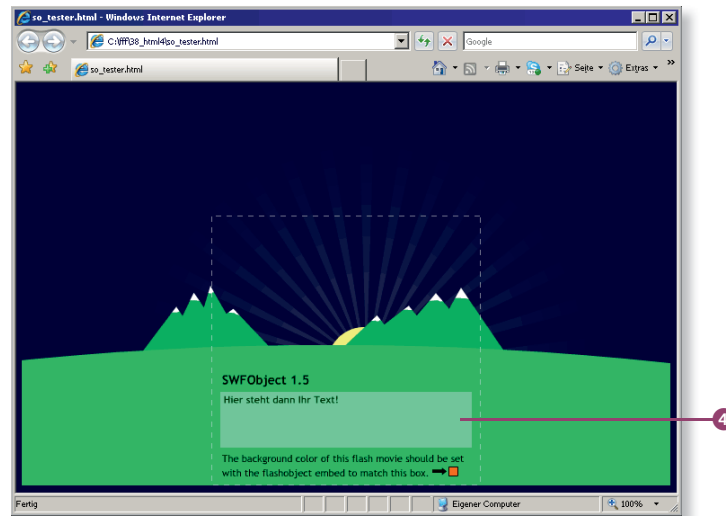


Zwischenergebnis:
37_KOMMUNIKATION
• SO_TESTER_STEP06.
HTML

Schritt 7

Im Browser testen

Abbildung 4 ►
Das Textfeld wird dynamisch gefüllt.



Schritt 8 Werte aus der URL auslesen

Zwischenergebnis:
37_KOMMUNIKATION
• SO_TESTER_STEP08.
HTML



Häufig werden beim Aufruf einer Webseite Variablen in der URL an die HTML-Seite mit übergeben. Hier ein Beispiel:

<http://www.ihredomain.de/index.html?id=8342>

Hier handelt es sich bei `id` um den Variablennamen und bei der Zahl hinter dem `=` um den Variablenwert. Wenn Sie nun diese Informationen aus HTML heraus an den Flash-Film weitergeben möchten, bietet Ihnen das SWFObject auch hier eine sehr elegante Lösung. Sie können über die `getQueryParamValue`-Funktion des SWFObjects direkt auf jeden Wert einer per URL übergebenen Variable zugreifen. Dazu müssen Sie lediglich den Namen dieser Variable kennen und der Funktion als Parameter übergeben. In dem oben angegebenen Beispiel würde das dann wie folgt aussehen:

```
so.addVariable("flashVarId", getQueryParamValue("id"));
```

Wenn Sie diese Zeile in Ihre html-Datei einfügen, können Sie innerhalb des Flash-Films über den Namen `flashVarId` auf die Variable zugreifen.

Öffnen Sie zu diesem Zweck ein weiteres Mal die im Schritt 6 geänderte `so_tester.html`, und fügen Sie die eben beschriebene Codezeile unterhalb der ersten `so.addVariable`-Anweisung ein.

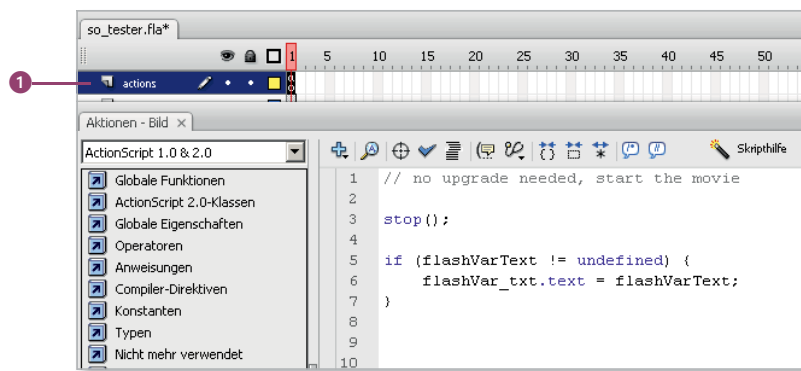
Speichern Sie die geänderte html-Datei als `so_tester.html` im aktuellen Workshop-Ordner ab.

Online testen

Wir bieten Ihnen hier bereits die Möglichkeit, sich das Ergebnis online einmal anzusehen, bevor Sie es nachbauen. So wird das Beispiel vielleicht etwas leichter verständlich:
<http://flashfastforward.de/swfobject/index.html?id=12345>

In Schritt 8 haben Sie dafür gesorgt, dass die an HTML übergebene Variable an den Flash-Film weitergegeben wird. Jetzt kümmern Sie sich darum, dass auch diese Variable in demselben Textfeld wie die `flashVarText`-Variable ausgegeben wird.

Hierzu öffnen Sie wieder die Datei `SO_TESTER.FLA` und lassen sich über `[F9]` bzw. am Mac über `[Alt] + [F9]` das AKTIONEN-FENSTER des ersten Bildes auf der Ebene »actions« 1 anzeigen.



Schritt 9

URL-Variable im Flash-Film anzeigen



Zwischenergebnis:
37_KOMMUNIKATION
• SO_TESTER_STEP09.
FLA

◀ **Abbildung 5**
ActionScript-Code der Ebene
»actions«

Fügen Sie unterhalb der ersten `if`-Anweisung eine zweite hinzu:

```
if (flashVarId != undefined) {
    flashVar_txt.text += " " + flashVarId;
}
```

Die zweite `if`-Anweisung überprüft, genau wie die erste, ob die abgefragte Variable bereits initialisiert wurde. Wenn dies zutrifft, wird der neue Variablenwert an das Textfeld angehängt.

Nachdem Sie die Änderungen vorgenommen haben, speichern Sie die Datei als `SO_TESTER.FLA`. Anschließend testen Sie den Flash-Film über STEUERUNG • FILM TESTEN oder den Tastaturbefehl `[Strg] / [⌘] + [↵]`.

Innerhalb der Flash-Umgebung sehen Sie noch immer den statischen Text. Dieser wird erst neu gesetzt, wenn Sie den Flash-Film über die HTML-Datei innerhalb des Browsers aufrufen.

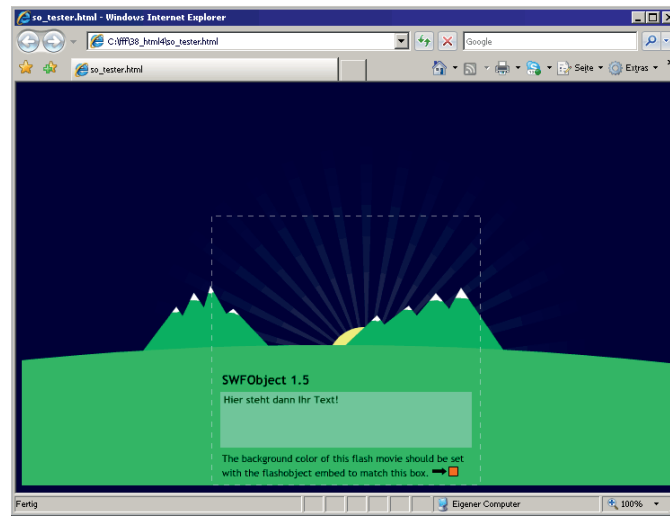
Schließen Sie das gerade geöffnete Flash-Player-Fenster wieder.

Wie eben beschrieben, werden Sie die Änderungen erst bei der Anzeige im Webbrowser sehen. Aus diesem Grund öffnen Sie die `SO_TESTER.HTML` im einem Browser Ihrer Wahl.

Schritt 10

Im Browser testen

Abbildung 6 ►
Das dynamische Textfeld zeigt weiterhin nur die erste Variable an.



Anwendungsgebiete

In der Praxis werden solche URL-Übergabeparameter z. B. oft bei Werbebannern benutzt: Man verlinkt auf die »normale« Flash-Seite, übergibt aber zugleich einen Parameter, so dass die Seite erkennen kann, dass der Besucher über ein Banner zu der Seite gelangt ist. Die Seite könnte so entsprechend personalisiert werden. Eine andere Möglichkeit der Anwendung sind Deeplinks. Diese führen den Besucher nicht zur Startseite, sondern, bei übergebenem Parameter, direkt in einen Unterbereich der Webseite.

Die `flashVarId` sehen Sie jedoch noch immer nicht innerhalb des Textfeldes – weil sie nämlich in der URL nicht übergeben wurde. Hierzu müssen Sie die URL in der Adresszeile des Browserfensters **1** um den folgenden String erweitern:

`?id=8342`

Nachdem Sie diesen Teil an die URL angehängt haben, bestätigen Sie mit der **↵**-Taste. Der Browser lädt die Datei neu, und die übergebene Variable erscheint im Textfeld **2**.

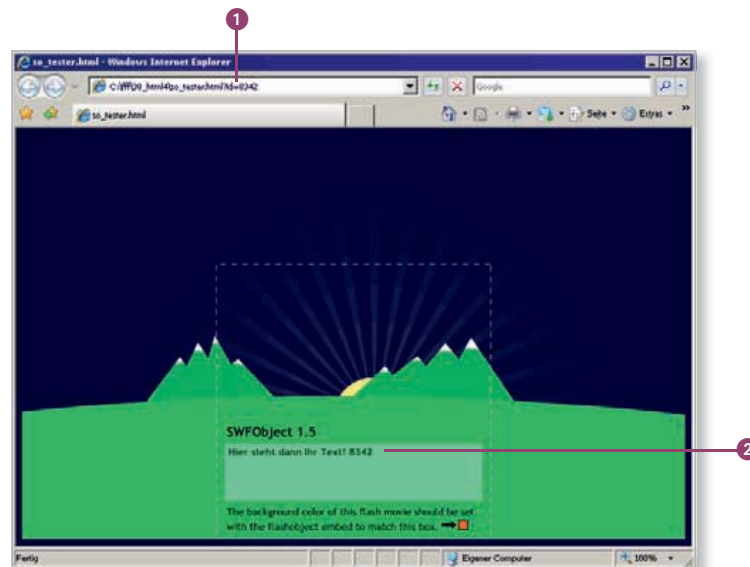


Abbildung 7 ►
Ihre per Hand eingetragene Variable wird im Flash-Film angezeigt.

Eine Website komplett in Flash

Deeplinks, Browsernavigation und benutzerfreundliche URLs

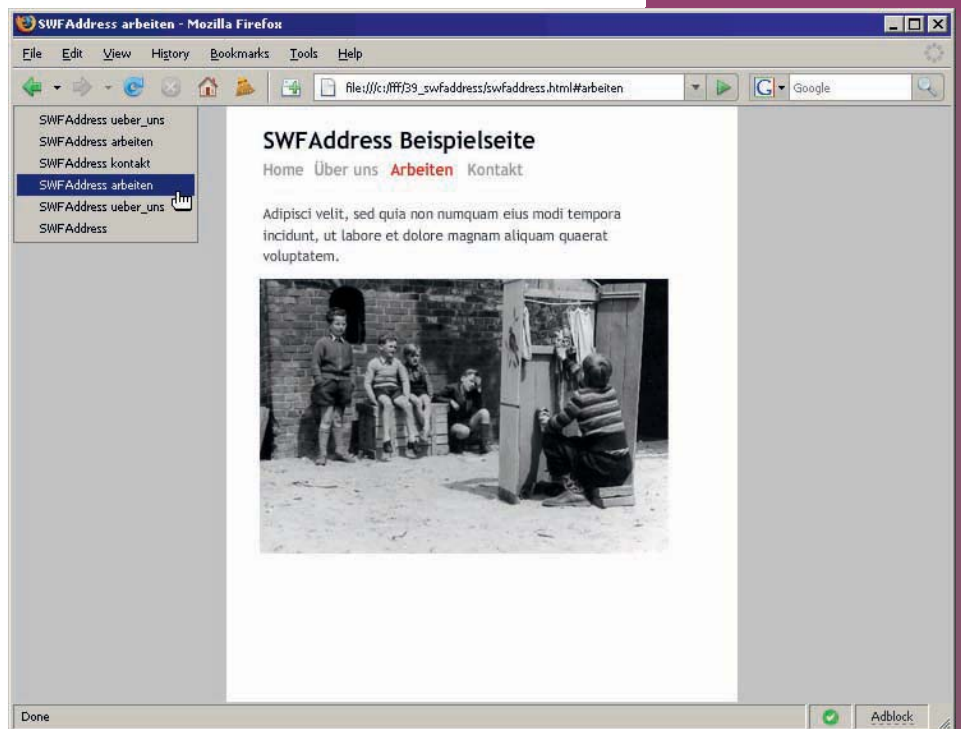
Sie wollten sicherlich auch schon einmal direkt auf eine Unterseite Ihrer Flash-Seite verlinken oder ein Zurückspringen über die Browsernavigation ermöglichen. Mit Hilfe des SWFAddress-Skripts, das auf dem SWFObject aufbaut, können Sie diese Funktionen sehr leicht implementieren. In diesem Workshop erlernen Sie den Umgang mit dem SWFAddress-Skript und wie Sie Ihre Seite damit aufwerten können.

Zielsetzungen:

Beispiel-Seite mit Flash-Navigation anlegen
SWFAddress integrieren
Browsernavigation in Flash abfangen

[Ergebnis: 38_WEBSITE-IN-FLASH • SWFADDRESS.HTML]

```
private function gotoPage(evt:MouseEvent):void
{
    switch(evt.target.name)
    {
        case "home_mc": SWFAddress.setValue(""); break;
        case "ueberUns_mc": SWFAddress.setValue("ueber_uns");
        break;
        case "arbeiten_mc": SWFAddress.setValue("arbeiten");
        break;
        case "kontakt_mc": SWFAddress.setValue("kontakt");
        break;
    }
}
```



Schritt 1 Vorbereitungen

Im ersten Schritt legen Sie einen neuen Ordner auf Ihrer Festplatte an. Nennen Sie ihn SWFADDRESS. Hier werden Sie die von Ihnen erstellten Beispiele zu diesem Workshop abspeichern.

Um diesen Workshop erfolgreich durchführen bzw. alle Schritte nachvollziehen zu können, sollten Sie bereits mit dem SWFObject aus den vorherigen Workshops vertraut sein.

Schritt 2 Ordnerstruktur anlegen

SWFObject

Eine Einführung in das SWF-Object erhalten Sie in den drei vorhergehenden Workshops. Diese beschäftigen sich mit den verschiedenen Optionen des SWFObjects.

Als Erstes kopieren Sie einige der benötigten Dateien in den eben erstellten Ordner. Da Sie in diesem Workshop mit vielen unterschiedlichen Dateiarten arbeiten werden, legen Sie folgende Ordner innerhalb des Workshop-Ordners an:

- ▶ ASSETS
- ▶ CLASSES
- ▶ JS

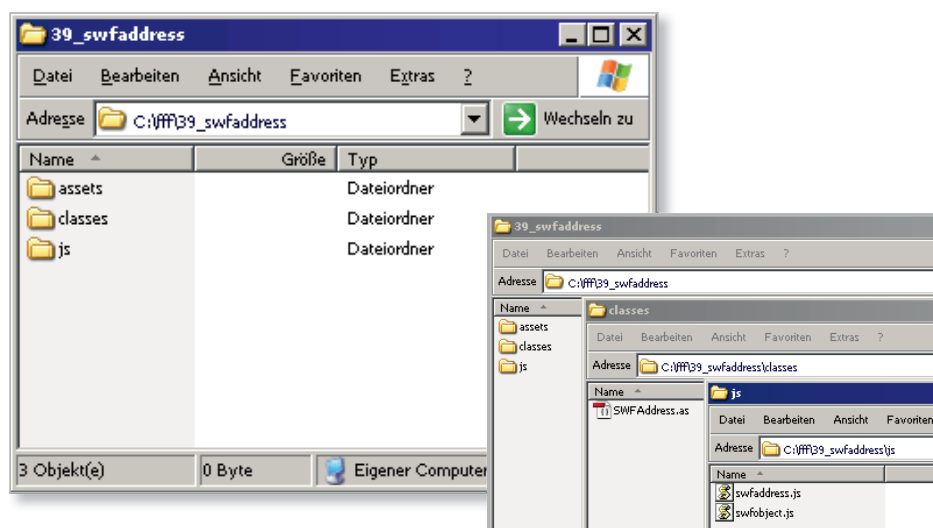


Abbildung 1 ▶

Die initiale Ordnerstruktur

Abbildung 2 ▶▶

Ordnerstruktur mit den anfangs benötigten Dateien

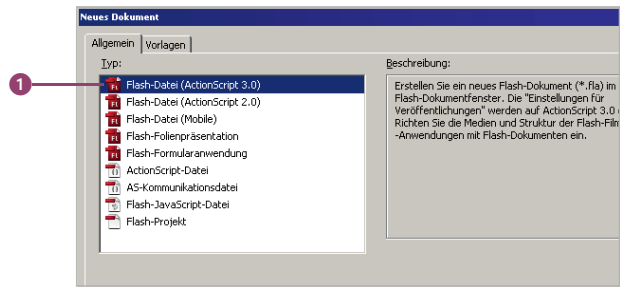
Schritt 3 Dateien kopieren

Im Ordner ASSETS werden Sie alle »Bauteile« für Ihren Flash-Film abspeichern. Der CLASSES-Ordner wird alle von Ihnen erzeugten Flash-Klassen enthalten. Und im Ordner JS werden Sie Ihre JavaScript-Dateien ablegen.

Kopieren Sie nun die benötigten Dateien SWFObject.JS und SWFADDRESS.JS aus dem zu diesem Workshop gehörenden Ordner der Buch-DVD in den JS-Ordner.

SWFADDRESS.AS, eine weitere für das SWFAddress benötigte Datei, speichern Sie im Ordner CLASSES.

Erstellen Sie dann eine FLASH-DATEI (ACTIONSCRIPT 3.0) ❶ innerhalb der Flash-Entwicklungsumgebung.

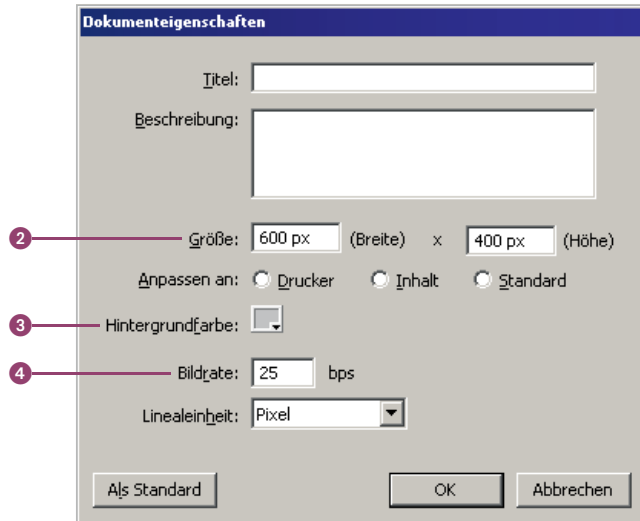


Schritt 4 Flash-Datei anlegen

◀ **Abbildung 3**
Eine neue Action-
Script 3.0-Datei anlegen

Speichern Sie die Datei über DATEI • SPEICHERN UNTER als SWFADDRESS.FLA direkt in Ihrem Workshop-Ordner 39_SWFADDRESS.

Ändern Sie nach dem Anlegen der Datei über MODIFIZIEREN • DOKUMENT die DOKUMENTEIGENSCHAFTEN. Die Grösse der Bühne soll 600 x 400 Pixel ❷ bei einer BILDRATE von 25 bps ❹ sein. Setzen Sie zudem die HINTERGRUNDFARBE ❸ auf ein helles Grau – Farbwert #CCCCCC. Die übrigen Eigenschaften lassen Sie unverändert. Bestätigen Sie mit OK.



Schritt 5 Dokumenteigenschaften anpassen

◀ **Abbildung 4**
Dokumenteigenschaften
festlegen

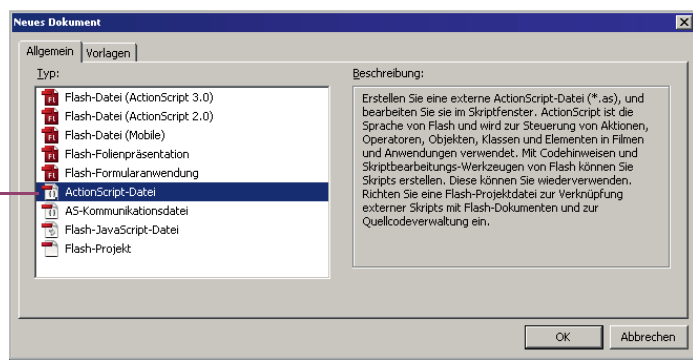
Dokumentklasse

Lesen Sie auf Seite 190 nach, was es mit der Dokumentklasse auf sich hat.

Nun erstellen Sie die Dokumentklasse. Sie wird in diesem Workshop die allgemeine Steuerung der Flash-Datei übernehmen. Erstellen Sie zu diesem Zweck mit dem Tastaturbefehl `[Strg]/[⌘] + [N]` eine neue ACTIONSCRIPT-DATEI ❺.

Schritt 6 Dokumentklasse erstellen

Abbildung 5 ▶
Erstellen einer ActionScript-Datei



Nachdem Sie die neue Datei geöffnet haben, speichern Sie sie über **[Strg]/[⌘] + [S]** als MAIN.AS im Unterordner CLASSES ab.

Schritt 7 Weitere Einstellungen ändern

Damit Flash die Datei MAIN.AS in dem von Ihnen erstellten Unterordner findet, müssen Sie noch die EINSTELLUNGEN FÜR VERÖFFENTLICHUNGEN anpassen. Hierzu öffnen Sie über **[Strg]/[Alt] + [⇧] + [F12]** das entsprechende Fenster und stellen als Erstes sicher, dass im Reiter FORMATE 1 nur bei FLASH (.SWF) 2 ein Häkchen gesetzt ist.

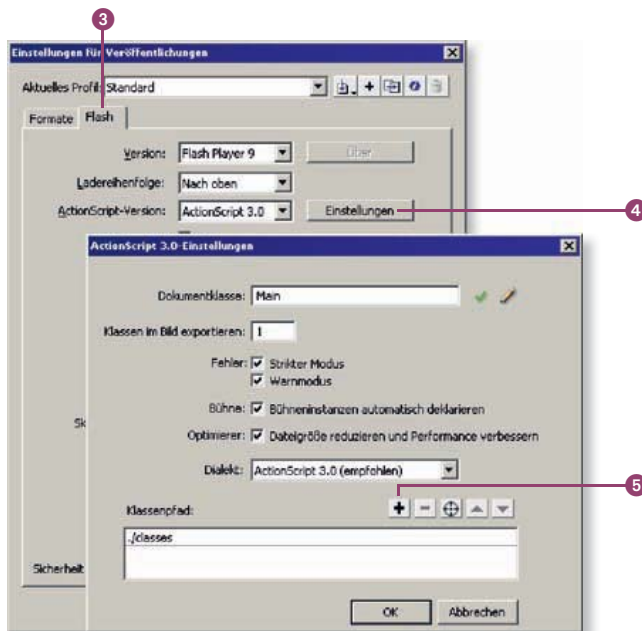
Abbildung 6 ▶
Aktivieren Sie das Häkchen
vor FLASH (.SWF).



Schritt 8 Klassenordner verknüpfen

Bevor Sie die Änderungen aus Schritt 7 mit OK bestätigen, wechseln Sie zum Reiter FLASH 3 und rufen das Untermenü ACTIONSCRIPT-3.0-EINSTELLUNGEN 4 auf. In dem neuen Fenster tragen Sie den KLASSENPFAD zu Ihrem CLASSES-Ordner ein. Klicken Sie hierzu auf den Plus-Button 5, und geben Sie dann »./classes« ein.

Nachdem Sie den Klassenpfad hinzugefügt haben, schließen Sie das Fenster mit OK und bestätigen alle in den Schritten 7 und 8 vorgenommenen Änderungen in den EINSTELLUNGEN FÜR VERÖFFENTLICHUNGEN wiederum mit OK.

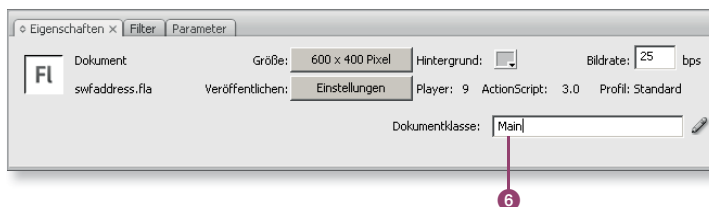


Klassenpfad korrekt angeben

Der Punkt vor `classes` steht für den aktuellen Ordner, in dem sich die `fla`-Datei befindet. Es ist immer gut, einen relativen und nicht einen absoluten Pfad zu dem Klassenordner anzugeben. So müssen Sie diese Einstellung nicht wieder ändern, wenn Sie den kompletten Workshop-Ordner später verschieben sollten.

◀ **Abbildung 7**
Klassenpfad angeben

Jetzt, da Sie den Klassenordner mit der Flash-Datei verknüpft haben, können Sie im Eigenschaftenfenster, das Sie über `[Strg]/[⌘] + [F3]` erreichen, die DOKUMENTKLASSE »Main« eintragen 6. Bestätigen Sie die Eingabe mit `[↵]`.



Schritt 9 Dokumentklasse festlegen

◀ **Abbildung 8**
Setzen der Dokumentklasse

Öffnen Sie die Datei `MAIN.AS` in Ihrem bevorzugten ActionScript 3-Editor, und fügen Sie folgende Zeilen hinzu:

```
1: package
2: {
3:     import flash.display.MovieClip;
4:     public class Main extends MovieClip
5:     {
6:         public function Main()
7:         {
8:         }
9:     }
10: }
```

Schritt 10 Grundgerüst von Main.as



Zwischenergebnis:
38_WEBSITE-IN-FLASH
• CLASSES • MAIN_
STEP10.AS

Tipp

Mehr zu den Themen »Packages« und »Vererbung« finden Sie im Exkurs »Klassen, Vererbung, Packages« in diesem Buch.

In Zeile 1 legen Sie das `package` fest, in dem sich die Klasse befindet. In diesem Fall gibt es kein `package`, aber diese Zeile müssen Sie trotzdem in jedem Fall mit angeben.

Die eigentliche Klassedefinition folgt in Zeile 4: Ihre Klasse heißt `Main` und erbt alle Eigenschaften von `MovieClip`. Die `MovieClip`-Klasse wird in Zeile 3 importiert.

In Zeile 6 implementieren Sie einen öffentlichen Konstruktor.

Speichern Sie die vorgenommenen Änderungen über `Strg`/`⌘`+`S`, und wechseln Sie wieder zur Datei `SWFADDRESS.FLA` in die Flash-Autorenumgebung.

Schritt 11 Seitenhintergrund anlegen

Den Hintergrund der Flash-Datei haben Sie bereits auf Grau gesetzt. Jetzt legen Sie eine weiße Fläche als Hintergrund für die Seiteninhalte an.

Benennen Sie die vorhandene Ebene in der Zeitleiste in »bg« um. Wählen Sie das Rechteckwerkzeug  aus der Werkzeugpalette, und zeichnen Sie ein beliebig großes Rechteck auf die Bühne. Anschließend wählen Sie das eben gezeichnete Rechteck mit dem Auswahlwerkzeug  aus und ändern im Eigenschaftenfenster die Werte für Position **2**, Größe **1** und Farbe **3**, so dass Sie danach ein 400x400 Pixel großes, weißes Rechteck haben, das auf der Position x:100 und y:0 liegt.

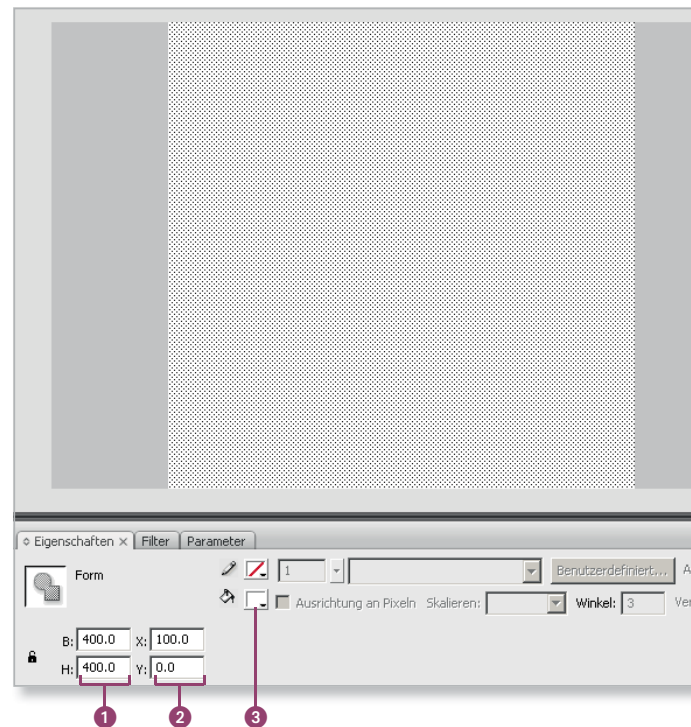
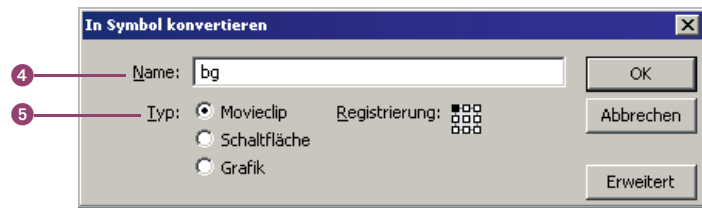


Abbildung 9 ►
Hintergrundgrafik anlegen

Dann konvertieren Sie über **[F8]** im Fenster **IN SYMBOL KONVERTIEREN** die ausgewählte Fläche in ein Symbol mit dem Namen »bg« **4**. Stellen Sie sicher, dass als **Typ** **5** **MOVIECLIP** ausgewählt ist, bevor Sie mit OK bestätigen.



Schritt 12

In Movieclip konvertieren

◀ **Abbildung 10**
In Movieclip konvertieren

Jetzt geben Sie dem Movieclip auf der Bühne den **INSTANZNAMEN** »bg_mc«.

Damit sich die weiße Fläche immer über die ganze Höhe des Flash-Films erstreckt, müssen Sie der Bühne einen **EventListener** hinzufügen, der »lauscht«, ob sich die Größe der Bühne ändert. Dies geschieht z. B., wenn der Besucher Ihrer Seite die Größe seines Browserfensters nachträglich verändert. Diese Anpassungen werden Sie im Konstruktor von **MAIN.AS** vornehmen, so dass sie gleich als Erstes nach Aufruf der **swf**-Datei ausgeführt werden.

Um diese Funktionalität zu implementieren, müssen Sie zuerst dafür sorgen, dass bestimmte Klassen innerhalb von **MAIN.AS** zur Verfügung stehen. Importieren Sie also unter der letzten Importanweisung folgende weitere Klassen:

```
import flash.display.StageAlign;
import flash.display.StageScaleMode;
import flash.events.Event;
```

Sobald dies geschehen ist, fügen Sie diese Zeilen innerhalb des Konstruktors hinzu:

```
1:    stage.align = StageAlign.TOP;
2:    stage.scaleMode = StageScaleMode.NO_SCALE;
3:    stage.addEventListener(Event.RESIZE, onResizeStage);
```

Die Zeilen 1 und 2 sorgen dafür, dass der Flash-Film mittig und am oberen Rand der **SWF** ausgerichtet und nicht skaliert wird, auch wenn die **swf**-Datei größer ist als die vorher festgelegten 600x400 Pixel.

Schritt 13

Hintergrund automatisch anpassen

In Zeile 3 fügen Sie der Bühne den erwähnten EventListener hinzu. Dieser Listener »lauscht« auf das `Resize`-Ereignis, das dann die `onResizeStage`-Funktion aufruft.

Schritt 14 EventHandler hinzufügen

Zwischenergebnis:
38_WEBSITE-IN-FLASH
• CLASSES • MAIN_
STEP14.AS



Der EventHandler wird am Ende der Klasse direkt unter dem Konstruktor eingefügt und sieht folgendermaßen aus:

```
1: private function onResizeStage(evt:Event=null):void
2: {
3:     bg_mc.height = stage.stageHeight;
4: }
```

Die Funktion ist `private` und erwartet als Übergabeparameter das gefeuerte `Event`. Sobald sich also die Bühnengröße verändert, wird diese Funktion aufgerufen. Sie passt in Zeile 3 die Höhe von `bg_mc` an die aktuelle Bühnenhöhe an.

Schritt 15 Flash-Film testen

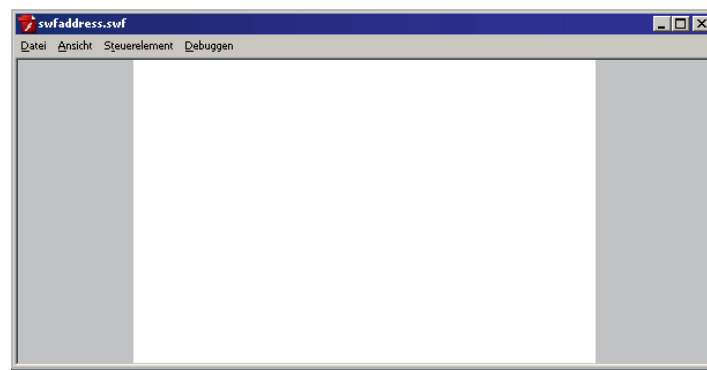
Tipp

Das `Resize`-Ereignis wird nur dann gefeuert, wenn Sie den `scaleMode` des Flash-Films auf `NO_SCALE` gesetzt haben.

Abbildung 11 ►

Der Hintergrund passt sich der Höhe des Flash-Player-Fensters automatisch an.

Testen Sie den Flash-Film über `[Strg]/[F5]+[↵]`. Verändern Sie dann die Größe des Flash-Player-Fensters. Der weiße Hintergrund sollte nun immer über die komplette Höhe des Fensters skaliert werden.



Schließen Sie den Flash Player wieder.

Schritt 16 Unterseiten erstellen

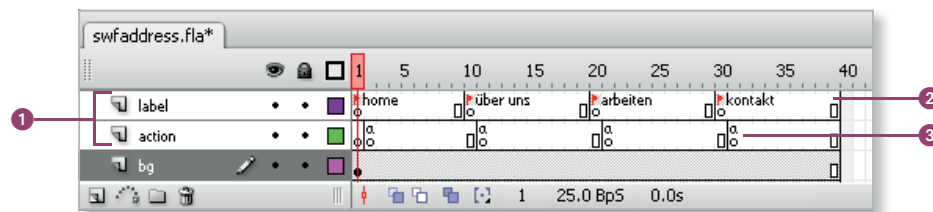
Um eine navigierbare Seite zu erhalten, müssen Sie nun einige Inhalte und Unterseiten erstellen.

Erzeugen Sie im ersten Schritt über `EINFÜGEN • ZEITLEISTE • EBENE` zwei weitere Ebenen auf der Zeitleiste, und nennen Sie diese »label« bzw. »ac-

tion« ❶. Auf der »label«-Ebene werden Sie die Sprungmarken für die Unterseiten erstellen. »action« beinhaltet alle Skripte, die Sie auf der Zeitachse platzieren werden – in diesem Fall ausschließlich `stop`-Befehle.

Nachdem Sie die Ebenen hinzugefügt haben, erstellen Sie vier Sprungmarken ❷ im Abstand von jeweils 10 Bildern: »home«, »über uns«, »arbeiten« und »kontakt«. Jede Sprungmarke wird auf der »action«-Ebene mit einem `stop()`; versehen ❸, so dass der Film nicht weiter abspielt, sobald die Marke einmal angesprungen wurde.

Ihre Zeitachse sieht nun folgendermaßen aus:



Tipp

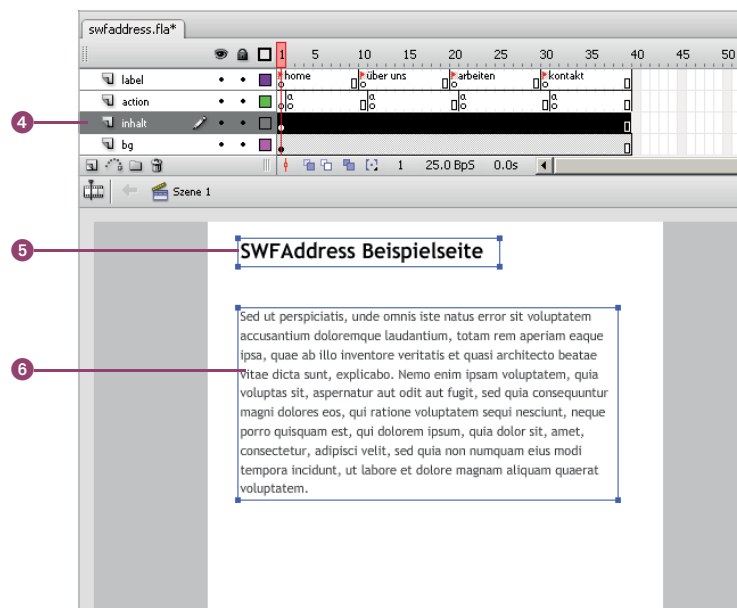
Platzieren Sie die `stop`-Anweisung immer im Bild rechts von der Sprungmarke (auf der »action«-Ebene). Dies verhindert, dass der Film weiter abspielt, sollte die Sprungmarke zweimal hintereinander mit `gotoAndPlay()` angesprungen werden.

◀ Abbildung 12

Die angepasste Hauptzeitachse

Momentan unterscheiden sich Ihre Seiten noch nicht voneinander – es fehlt der Inhalt der jeweiligen Unterpunkte. Um diese zu befüllen, erstellen Sie eine weitere Ebene und nennen sie »inhalt« ❹.

Als Erstes legen Sie mit Hilfe des Textwerkzeugs **T** einen Titel für Ihre Seite ❺ fest. Dieser sollte ungefähr auf der Position x:126 und y:14 liegen. Zusätzlich fügen Sie einen etwas längeren Text ❻ hinzu. Dieser Inhalt wird sich von Seite zu Seite unterscheiden.



Schritt 17

Die erste Unterseite befüllen

◀ Abbildung 13

Die erste Seite ist fertig.

Schritt 18

Die restlichen Unterseiten abändern

Um die weiteren Seiten zu befüllen, erstellen Sie jeweils unterhalb der einzelnen Sprungmarken mit **[F6]** auf der Ebene »inhalt« drei neue Schlüsselbilder **1**. Ändern Sie den Text auf den Seiten so, dass sich jedes Schlüsselbild von den anderen unterscheidet.

Die Unterseite »arbeiten« erhält zur weiteren Unterscheidung ein Foto (JPG, GIF o.Ä.). Importieren Sie hierzu über **[Strg]/[⌘]+[R]** eine Datei Ihrer Wahl direkt auf die Bühne. Skalieren Sie das Foto anschließend mit dem Frei-Transformieren-Werkzeug **[F]** so, dass es innerhalb der weißen Fläche Platz findet.



Abbildung 14 ►
Auch ein Bild darf nicht fehlen.

Schritt 19

Flash-Film testen

Speichern Sie die Änderungen mit **[Strg]/[⌘]+[S]** ab. Anschließend testen Sie den Film über **[Strg]/[⌘]+[↵]**. Wenn Sie innerhalb des Flash Players **[↵]** drücken, spielt der Film bis zum nächsten **stop()** – so können Sie schon jetzt die einzelnen Unterseiten betrachten.



Abbildung 15 ►
Die vier Unterseiten im Überblick

Damit sich ein Besucher auf Ihrer Flash-Seite bewegen kann, erzeugen Sie in diesem Schritt eines der benötigten Menüelemente. Die Menüpunkte sollen Textelemente sein, die im aktiven Zustand bzw. beim Darüberfahren mit der Maus die Farbe wechseln.

Da das Menü auf allen Unterseiten zu sehen sein wird, legen Sie eine weitere Ebene mit dem Namen »menu« an. Diese liegt oberhalb von »inhalt« und wird nur ein Schlüsselbild enthalten ❷. So stellen Sie sicher, dass sich das Menü nicht versehentlich von Seite zu Seite verschiebt.

Danach erstellen Sie den »home«-Menüpunkt unterhalb des Seitentitels. Hierzu wählen Sie das Textwerkzeug **T** aus und erzeugen ein Textfeld mit dem Inhalt »Home« auf der Bühne ❸. Im Anschluss konvertieren Sie das markierte Textfeld über **[F8]** in einen Movieclip und geben ihm den Namen »btn home« ❹.

Als Instanzname wählen Sie »home_mc«.



Schritt 20

Menü-Element erzeugen

◀ Abbildung 16

In Symbol konvertieren

Die einzelnen Menüpunkte sollen beim Darüberfahren mit der Maus die Farbe wechseln. Um dies zu erreichen, muss jeder Menüpunkt über zwei unterschiedliche Zustände verfügen.

Wechseln Sie nun in den erstellten Movieclip, und benennen Sie die hier bereits vorhandene Ebene in »text« um. Erzeugen Sie zudem zwei neue Ebenen »label« und »action« ❺. Im nächsten Schritt legen Sie zwei Sprungmarken »_up« und »_over« an ❻ und fügen zu jeder Marke auf der Ebene »action« eine **stop**-Anweisung hinzu ❼.

Im ersten Bild von »_over« erstellen Sie auf der Ebene »text« ein Schlüsselbild, wählen das Textfeld aus und ändern die Textfarbe in #CC0000 ❽.

Schritt 21

Rollover-Menü vorbereiten

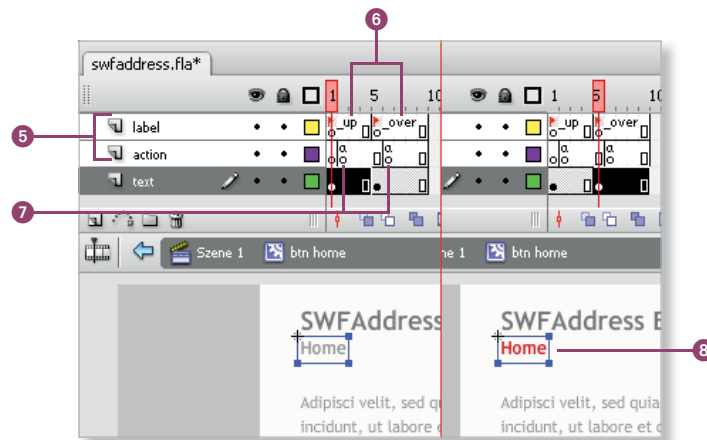
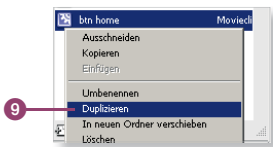


Abbildung 17 ►
Den Rollover-Effekt anlegen

Schritt 22 Menüelement duplizieren



▲ Abbildung 18
Duplizieren der Navigations-
elemente

Schritt 23 Menüpunkte auf der Bühne platzieren

Um sich die Erstellung der anderen Menüpunkte zu vereinfachen, duplizieren Sie den eben angelegten Movieclip. Hierzu wählen Sie ihn in der Bibliothek aus, drücken die rechte Maustaste und wählen den Menüpunkt **DUPlizIEREN** 9.

Führen Sie diesen Schritt dreimal durch, und benennen Sie die jeweils neu entstandenen Movieclips mit »btn über uns«, »btn arbeiten« und »btn kontakt«. Anschließend öffnen Sie diese und ändern den Text entsprechend der Movieclip-Namen in »Über uns«, »Arbeiten« bzw. »Kontakt«.

Ziehen Sie nun die drei neuen Menüpunkte aus der Bibliothek auf die Bühne, und platzieren Sie sie neben dem bereits vorhandenen »Home«-Menüpunkt auf der »menu«-Ebene 10.

Vergeben Sie zudem folgende INSTANZNAMEN an die Movieclips: »ueber-Uns_mc«, »arbeiten_mc« sowie »kontakt_mc«.

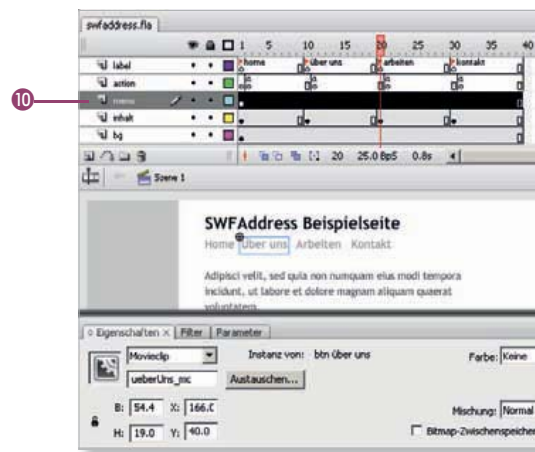
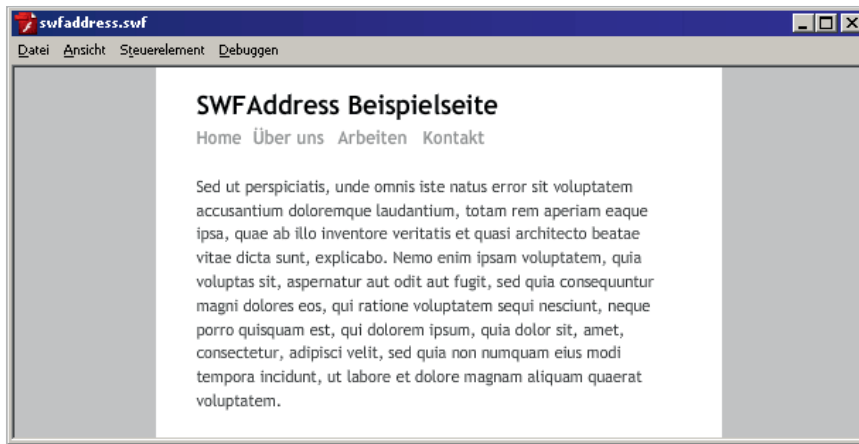


Abbildung 19 ►
Platzierung am Beispiel des
»Über uns«-Movieclips

Speichern Sie die vorgenommenen Änderungen mit dem Tastaturbefehl `Strg`/`⌘` + `S`.

Über **STEUERUNG • FILM TESTEN** öffnen Sie den aktuellen Film im Flash Player der Entwicklungsumgebung.

Sie sehen nun alle Menüpunkte – diese verhalten sich aber noch nicht wie solche: Man kann weder zwischen den Seiten hin- und herspringen, noch wechseln die Menüpunkte die Farbe, wenn man mit der Maus darüberfährt.



Schritt 24

Flash-Film testen

◀ **Abbildung 20**
So sieht Ihr Menü aus.

In diesem Schritt werden Sie sich als Erstes um die aktiven Zustände der einzelnen Menüpunkte kümmern. Danach implementieren Sie die eigentliche Seitensteuerung mit Hilfe des SWFAddress-Skripts.

Öffnen Sie die Datei **MAIN.AS** in Ihrem Editor, und fügen Sie oberhalb des Konstruktors zwei Klasseneigenschaften hinzu. Die erste, `menu_arr`, ist ein Array und beinhaltet die Namen aller Menüpunkte auf der Bühne. Sie ist `private`, da sie nur innerhalb der Klasse benötigt wird.

```
private var menu_arr:Array = ["home_mc", "ueberUns_mc",  
"arbeiten_mc", "kontakt_mc"];
```

Als zweite Eigenschaft deklarieren Sie eine Variable des Typs `uint` mit dem Namen `currentIndex`. Auch diese Eigenschaft ist `private`. Sie dient der Speicherung des aktuell ausgewählten Menüpunkt-Indexes.

```
private var currentIndex:uint;
```

Da »Home« die erste Unterseite ist, die aufgerufen wird, wenn Sie den Flash-Film starten, setzen Sie innerhalb des Konstruktors den Wert der soeben deklarierten Variable `currentIndex` auf 0.

Schritt 25

Dem Menü Leben einhauchen



Zwischenergebnis:
38_WEBSITE-IN-FLASH
• CLASSES • MAIN_
STEP25.AS

```
currentIndex = 0;
```

Jetzt fehlt noch die Funktion, die den Rollover-Status der Menüpunkte umsetzt. Schreiben Sie diese unterhalb des Konstruktors:

```
1: private function initMenu():void
2: {
3:     for(var i:uint=0;i<menu_arr.length;i++)
4:     {
5:         if(i == currentIndex)
6:         {
7:             this[menu_arr[i]].buttonMode = false;
8:             this[menu_arr[i]].gotoAndPlay("_over");
9:         }
10:        else
11:        {
12:            this[menu_arr[i]].buttonMode = true;
13:            this[menu_arr[i]].gotoAndPlay("_up");
14:        }
15:    }
16: }
```

buttonMode

Ist der `buttonMode` eines Movieclips auf `true` gesetzt, verhält sich dieser wie ein Button und zeigt bei Rollover der Maus einen Hand-Cursor an. Wenn der Movieclip die Sprungmarken »_up«, »_over« bzw. »_down« beinhaltet, wird zudem die jeweilige Sprungmarke angesprungen.

Innerhalb der `initMenu`-Methode wird in Zeile 3 das komplette `menu_arr`-Array durchlaufen. Dabei wird der `buttonMode` des aktiven Menüpunktes auf `false` gesetzt (Zeile 7) und der des inaktiven auf `true` (Zeile 12).

Weiterhin wird in Zeile 8 der jeweils aktive Punkt hervorgehoben bzw. in Zeile 13 bei Inaktivität diese Hervorhebung wieder aufgehoben.

Nun müssen Sie die Funktion noch aufrufen. Ergänzen Sie dazu in der letzten Zeile des Konstruktors folgende Anweisung:

```
initMenu();
```

Speichern Sie mit `Strg/⌘+S` die Änderungen in `MAIN.AS`.

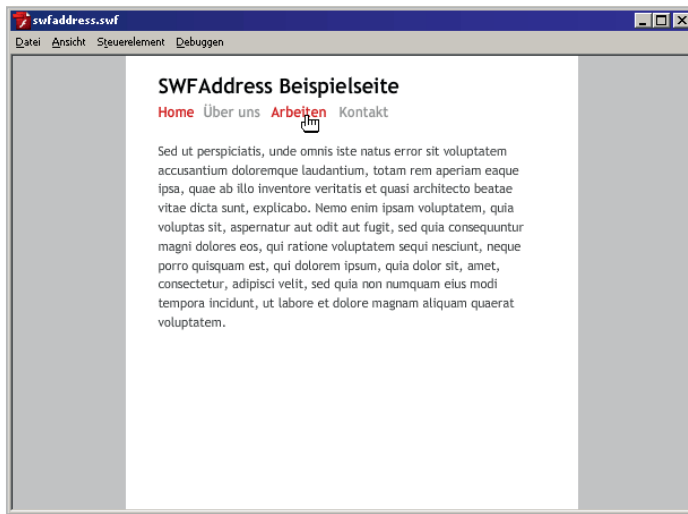
Schritt 26

Flash-Film testen

Testen Sie den Flash-Film mit dem Tastaturbefehl `Strg/⌘+↵`.

Sie können sehen, dass der erste Menüpunkt »Home« rot hervorgehoben ist. Fahren Sie mit der Maus über die anderen Menüpunkte. Diese sollten nun beim Rollover auch rot eingefärbt werden.

Der Seitenwechsel bei einem Klick auf die Menüpunkte funktioniert jedoch noch nicht.



◀ **Abbildung 21**

Es passiert schon etwas ...

Um den Klick auf die Menüpunkte abzufangen, benötigen Sie Zugriff auf die `MouseEvent`-Klasse. Importieren Sie diese am Anfang Ihrer Klasse unter den in Schritt 13 eingefügten Import-Anweisungen:

```
import flash.events.MouseEvent;
```

Jetzt fügen Sie jedem Menüpunkt einen eigenen `EventListener` hinzu, der genau diese Klick-Events abfängt bzw. weiterleitet. Zu diesem Zweck erweitern Sie die `initMenu`-Methode um zwei Zeilen. Fügen Sie die folgende Zeile in die letzte Zeile der `if`-Anweisung ein:

```
this[menu_arr[i]].removeEventListener(MouseEvent.CLICK, gotoPage);
```

Ergänzen Sie dann entsprechend in der letzten Zeile der `else`-Anweisung folgenden Code:

```
this[menu_arr[i]].addEventListener(MouseEvent.CLICK, gotoPage);
```

Die zweite Zeile fügt den nicht aktiven Menüpunkten den `EventListener` hinzu; in der ersten Zeile wird dieser, sollte der Menüpunkt aktiv sein, wieder entfernt.

Das `CLICK`-Ereignis leiten Sie an die bisher noch fehlende Methode `gotoPage()` weiter. Diese Methode werden Sie im nächsten Schritt definieren.

Schritt 27

EventListener hinzufügen



Zwischenergebnis:
38_WEBSITE-IN-FLASH
• CLASSES • MAIN_
STEP27.AS

Schritt 28

SWFAddress integrieren

Zwischenergebnis:
38_WEBSITE-IN-FLASH
• CLASSES • MAIN_
STEP28.AS



Switch-Anweisung

In diesem Fall lösen Sie die Aufgabe mit Hilfe einer `switch`-Anweisung. Sie hätten genauso gut auch mehrere `if`-/else-Anweisungen verwenden können. Unser Lösungsansatz ist aber übersichtlicher und lässt sich in weniger Codezeilen umsetzen.

Hinweis

Da die SWFAddress-Klasse in demselben Package wie die MAIN.AS-Datei liegt, müssen Sie die Klasse nicht extra importieren, um sie benutzen zu können.

Bei einer »ganz normalen« Flash-Seite würden Sie jetzt innerhalb der `gotoPage`-Methode die Seitensteuerung umsetzen. In diesem Workshop wollen Sie aber das SWFAddress-Skript in Ihre Seite einbinden. Deshalb müssen Sie einen kleinen Umweg wählen.

Fügen Sie die neue Methode oberhalb der `initMenu`-Funktion in Ihre MAIN.AS-Datei ein.

```
1: private function gotoPage(evt:MouseEvent):void
2: {
3:     switch(evt.target.name)
4:     {
5:         case "home_mc": SWFAddress.setValue(""); break;
6:         case "ueberUns_mc": SWFAddress.setValue("ueber_uns");
7:         break;
8:         case "arbeiten_mc": SWFAddress.setValue("arbeiten");
9:         break;
10:        case "kontakt_mc": SWFAddress.setValue("kontakt");
11:        break;
12:        default: SWFAddress.setValue("");
13:    }
14: }
```

Bei einem Klick auf einen inaktiven Menüpunkt wird die Methode `gotoPage()` aufgerufen und ihr ein `MouseEvent` mit übergeben. Diesem Event entnehmen Sie in Zeile 3, welcher Menüpunkt das Event gefeuert hat, und verarbeiten diese Information innerhalb der `switch`-Anweisung.

Für jeden der möglichen Menüpunkte wird der Wert der SWFAddress-Klasse per `setValue()` auf einen passenden Namen gesetzt (Zeilen 5 bis 12).

Der `default`-Block der `switch`-Anweisung (Zeile 12) wird nur dann aufgerufen, wenn keiner der anderen Fälle zutrifft. Ist dies gegeben, setzen Sie den Wert genau wie in Zeile 5 auf einen leeren String.

Schritt 29

Seitenwechsel erreichen

Im vorherigen Schritt haben Sie sichergestellt, dass bei der Auswahl eines Menüpunkts der Wert der SWFAddress-Klasse angepasst wird. Den eigentlichen Seitenwechsel haben Sie damit allerdings noch nicht erreicht.

Nachdem Sie den Wert neu gesetzt haben, ruft die SWFAddress-Klasse die Funktion `SWFAddress.onChange` auf. Genau diesen Aufruf werden Sie an eine Methode innerhalb von MAIN.AS weiterleiten. Hierzu fügen Sie Ihrem Konstruktor in der ersten Zeile eine weitere Anweisung hinzu:

```
SWFAddress.onChange = onChange;
```

Mit dieser Zeile erreichen Sie, dass, wenn `SWFAddress.onChange()` aufgerufen wird, dieser Aufruf an die noch zu definierende `onChange`-Methode innerhalb Ihrer Main-Klasse weitergegeben wird.

Fügen Sie die folgende `onChange`-Methode unter der `initMenu`-Methode Ihrer `MAIN.AS` hinzu.

```
1: private function onChange():void
2: {
3:     var address:String = SWFAddress.getValue();
4:     switch(address)
5:     {
6:         case "": currentIndex = 0; gotoAndPlay("home");
7:         break;
8:         case "ueber_uns": currentIndex = 1; gotoAndPlay("über
9:         uns"); break;
10:        case "arbeiten": currentIndex = 2;
11:        gotoAndPlay("arbeiten"); break;
12:        case "kontakt": currentIndex = 3;
13:        gotoAndPlay("kontakt"); break;
14:        default: currentIndex = 0; gotoAndPlay("home");
15:    }
16:    initMenu();
17: }
```

In Zeile 3 holen Sie sich mit `getValue()` den aktuellen Wert der `SWF-Address`-Klasse und speichern ihn in der temporären Variable `address`. Anhand dieses Strings entscheiden Sie in Zeile 4 mit Hilfe einer `switch`-Anweisung, welcher Menüpunkt aktiv ist (`currentIndex`) bzw. zu welcher Unterseite innerhalb des Flash-Films gesprungen werden soll (`gotoAndPlay`) (Zeilen 6 bis 12). Sollte keiner der aufgeführten Fälle zutreffen, setzen Sie die »Home«-Unterseite in Zeile 14 als aktive Seite.

Nachdem die aktive Seite ermittelt wurde, rufen Sie in Zeile 16 ein weiteres Mal die `initMenu`-Methode auf, die das angezeigte Menü anpasst (siehe Schritt 25).

Speichern Sie mit `Strg`/`⌘` + `S` Ihre Änderungen in der Datei `MAIN.AS`.



Zwischenergebnis:
38_WEBSITE-IN-FLASH
• CLASSES • MAIN_
STEP29.AS

Schritt 30

onChange-Methode implementieren



Zwischenergebnis:
38_WEBSITE-IN-FLASH
• CLASSES • MAIN_
STEP30.AS

Schritt 31

Flash-Film testen

Wenn Sie nun den Flash-Film über `Strg/⌘ + ⬅` testen, können Sie zwischen den Seiten hin- und herspringen.

Um aber die volle Funktionalität der SWFAddress-Klasse sehen zu können, müssen Sie die Seite im Webbrowser testen. Hierzu fehlt Ihnen bislang noch die erforderliche HTML-Datei. Diese werden Sie im nächsten Schritt anlegen.

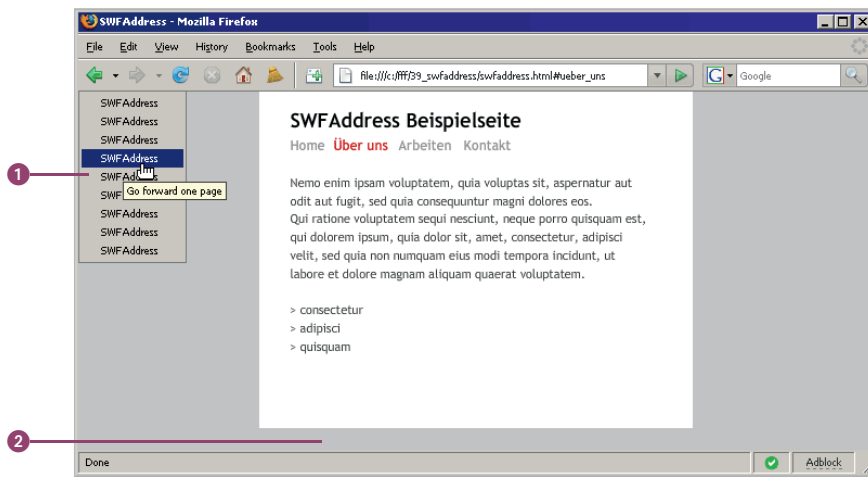
Schritt 32

HTML-Datei erzeugen

Öffnen Sie Ihren HTML- bzw. Text-Editor, und erstellen Sie eine neue html-Datei mit dem Namen SWFADDRESS.HTML. Speichern Sie diese im aktuellen Workshop-Ordner auf der obersten Ebene ab.

Binden Sie Ihren Flash-Film mit Hilfe des SWFObject in die html-Seite ein. Binden Sie hierzu folgende Zeilen JavaScript in den `<body>` der Datei ein:

```
<script type="text/javascript">
// 
var so = new SWFObject("swfaddress.swf", "sotest", "100%",
"100%", "9.0.45", "#CCCCCC");
so.useExpressInstall("expressinstall.swf");
so.write("flashcontent");
// ]]&gt;
&lt;/script&gt;</pre></div><div data-bbox="270 546 866 605" data-label="Text"><p>Jetzt fehlt lediglich noch die Integration des SWFAddress-JavaScripts. Zu diesem Zweck laden Sie im Anschluss an die SWFObject.js im <code>&lt;head&gt;</code>-Bereich die SWFADDRESS.js wie folgt:</p></div><div data-bbox="270 625 851 642" data-label="Text"><pre>&lt;script type="text/javascript" src="js/swfaddress.js"&gt;&lt;/script&gt;</pre></div><div data-bbox="270 665 776 685" data-label="Text"><p>Speichern Sie die Änderungen in Ihrer SWFADDRESS.HTML-Datei.</p></div><div data-bbox="270 686 866 722" data-label="Text"><p>Alternativ benutzen Sie die mitgelieferte html-Datei von der Buch-DVD.</p></div><div data-bbox="88 764 246 802" data-label="Section-Header"><hr/><h3>Schritt 33</h3><h4>Im Browser testen</h4></div><div data-bbox="270 764 693 782" data-label="Text"><p>Öffnen Sie die eben erstellte html-Datei im Browser.</p></div><div data-bbox="270 784 866 883" data-label="Text"><p>Navigieren Sie mit Hilfe des Menüs durch die Seiten. Ab sofort können Sie auch die Vor- und Zurück-Buttons des Browsers benutzen, um die bereits angesehenen Unterseiten aufzurufen. Zudem funktioniert der Aktualisieren-Button bzw. <code>F5</code> so, dass Sie nach erfolgreichem Neuladen der Seite auf der Unterseite landen, auf der Sie sich vorher befunden haben.</p></div><div data-bbox="46 936 212 954" data-label="Page-Footer"><p>450 Flash im Internet</p></div>
```



▲ **Abbildung 22**
Über die Browser-Buttons navigieren

Es fällt Ihnen vielleicht auf, dass zum einen alle angesehenen Unterseiten denselben Namen haben ❶ und dass zum anderen die Fläche des weißen Hintergrunds anfangs nicht über die komplette Browserhöhe geht ❷. Erst nachdem Sie die Größe des Browserfensters geändert haben, passt sich der Hintergrund der Höhe des Fensters an.

Diese beiden Probleme werden Sie in den nächsten Schritten noch beheben.

Die SWFAddress-Klasse verfügt über die Funktion `setTitle()`, mit der Sie den Titel der html-Seite laufend anpassen können.

Öffnen Sie hierzu ein weiteres Mal die Datei `MAIN.AS` in Ihrem ActionScript-Editor, und fügen Sie folgende Zeile in die `onChange`-Methode oberhalb der `switch`-Anweisung ein.

```
SWFAddress.setTitle("SWFAddress " + address);
```

Mit diesem Aufruf setzen Sie den HTML-Titel bei jedem Seitenwechsel neu.

Damit der weiße Hintergrund nicht erst nach Verändern der Größe des Browserfensters die richtige Höhe erhält, fügen Sie dem Konstruktor über dem Aufruf von `initMenu` eine weitere Codezeile hinzu.

Lokaler Test

Für den lokalen Test bietet sich der Webbrowser Mozilla Firefox an, da er auch lokal alle Funktionen des SWFAddress-Skripts problemlos ausführen kann. Dagegen funktionieren im Internet Explorer 7 z. B. der Vor- und Zurück-Button des Browsers nicht. Wenn Sie die Funktionen im Internet Explorer testen wollen, müssen Sie die erstellten Dateien zuerst auf einen Webserver hochladen.

Schritt 34 Seitentitel nachträglich ändern



Zwischenergebnis:
38_WEBSITE-IN-FLASH
• CLASSES • MAIN_
STEP34.AS

Schritt 35 Hintergrund korrigieren

Zwischenergebnis:
38_WEBSITE-IN-FLASH
• CLASSES • MAIN_
STEP35.AS



```
onResizeStage();
```

Diese Zeile bewirkt, dass gleich am Anfang des Flash-Films einmal die `onResizeStage`-Methode aufgerufen wird. Diese sorgt dafür, dass die Höhe des Flash-Films im Browser ausgelesen und der Hintergrund an diese Höhe angepasst wird.

Speichern Sie mit `Strg`/`⌘` + `S` Ihre Änderungen in `MAIN.AS`, und öffnen Sie anschließend die Datei `SWFADDRESS.FLA`. Diese müssen Sie jetzt noch über `⌘` + `F12` veröffentlichen, damit die Änderungen der `MAIN.AS` auch in der neuen `swf`-Datei wirksam werden.

Schritt 36 Im Browser testen

Wenn Sie Ihre Datei `SWFADDRESS.HTML` nun erneut im Webbrowser testen, stellen Sie fest, dass beim Seitenwechsel der HTML-Titel der Seite angepasst wird ❶.

Zudem sollte das Problem mit dem Hintergrund gelöst sein.

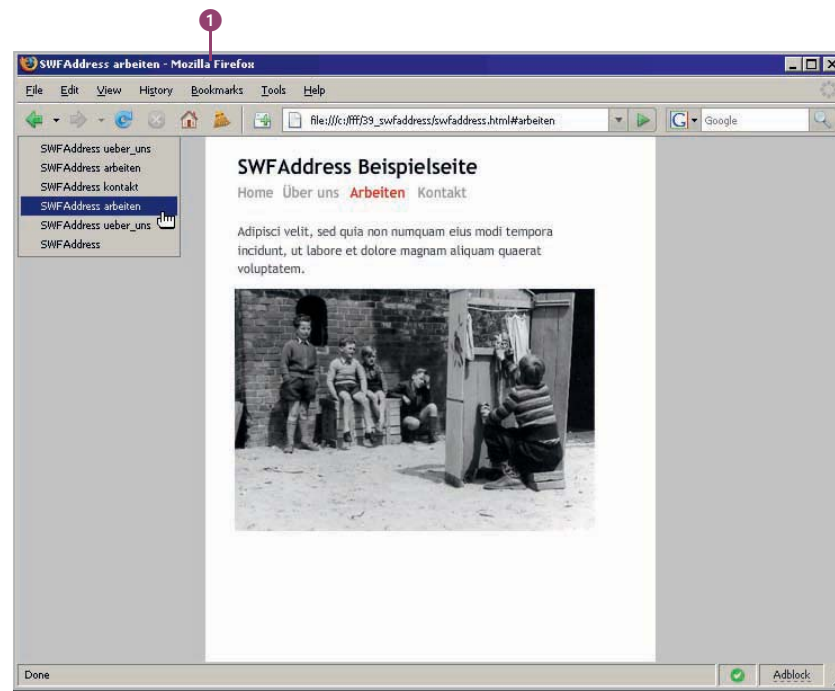


Abbildung 23 ►
Fertig ist Ihre Flash-Seite!

Das fertige Beispiel können Sie sich unter <http://www.flashfastforward.de/swfaddress> anschauen.

Skype-Status

Den aktuellen Skype-Status auf Ihren Webseiten integrieren

Skype – der sehr weit verbreitete Instant-Messenger/Chat-Client – bietet Usern die Möglichkeit, ihren Online-Status auf z. B. ihren eigenen Webseiten anzuzeigen. In diesem Workshop werden Sie es den Besuchern Ihrer Webseiten ermöglichen, Ihren Skype-Status zu erfragen und Sie direkt zu kontaktieren – sei es per Anruf oder Chat – oder einfach nur, um Sie in ihre Kontaktliste aufzunehmen.

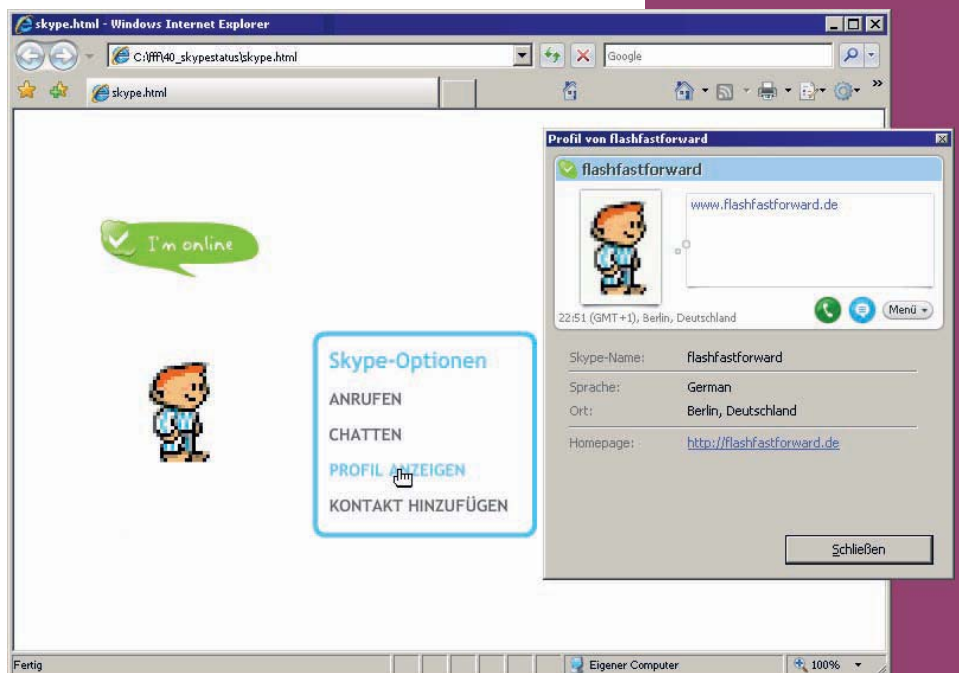
Zielsetzungen:

- Skype-Status erfragen
- Skype konfigurieren
- Status anzeigen
- Skype-Funktionen integrieren

[Ergebnis: 39_SKYPE-STATUS

• SKYPE.HTML]

```
import flash.display.Loader;
import flash.net.URLRequest;
import flash.events.MouseEvent;
import flash.external.ExternalInterface;
var mySkypeUsername:String = "flashfastforward";
var loader:Loader = new Loader();
loader.x = 10;
loader.y = 10;
loader.load(new URLRequest("http://mystatus.skype.com/balloon/"+mySkypeUsername));
addChild(loader);
if(ExternalInterface.available &&
!Boolean(ExternalInterface.call("skypeCheck"))){
{
```

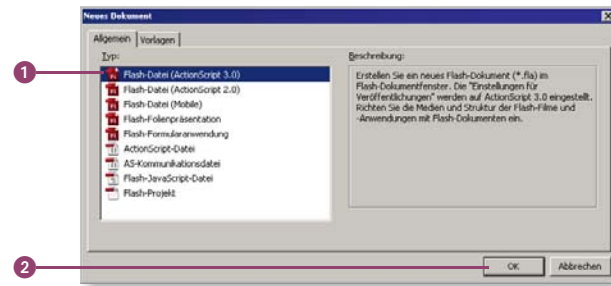


Schritt 1 Vorbereitungen

Im ersten Schritt legen Sie einen neuen Ordner auf Ihrer Festplatte an und nennen ihn SKYPESTATUS. Hier werden Sie die von Ihnen erstellten Beispiele zu diesem Workshop speichern.

Schritt 2 Flash-Datei erzeugen

Starten Sie die Flash-Autorenumgebung, und öffnen Sie über **[Strg]/[N]** den Dialog NEUES DOKUMENT. Hier wählen Sie als Dateityp FLASH-DATEI (ACTIONSCRIPT 3.0) ❶ und bestätigen mit **OK** ❷.



▲ **Abbildung 1**
Neue Flash-Datei anlegen (ActionScript 3.0)

Die neue Datei speichern Sie über **DATEI • SPEICHERN UNTER** unter dem Namen SKYPE.FLA im in Schritt 1 erstellten Ordner.

Schritt 3 Dokumenteigenschaften anpassen

Anschließend öffnen Sie über **[Strg]/[N] + [J]** die DOKUMENTEIGENSCHAFTEN und passen die Werte für die GRÖSSE der Bühne ❷ auf 400x300 Pixel an. Die BILDRATE ist in diesem Fall unwichtig – belassen Sie sie bei 12 bps. Bestätigen Sie die Änderungen mit **OK**.

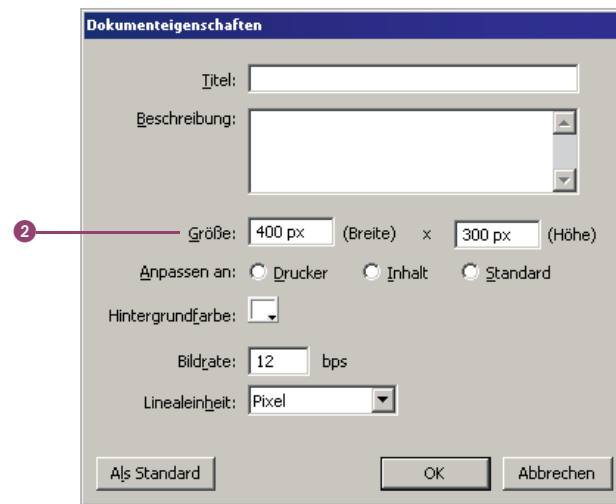
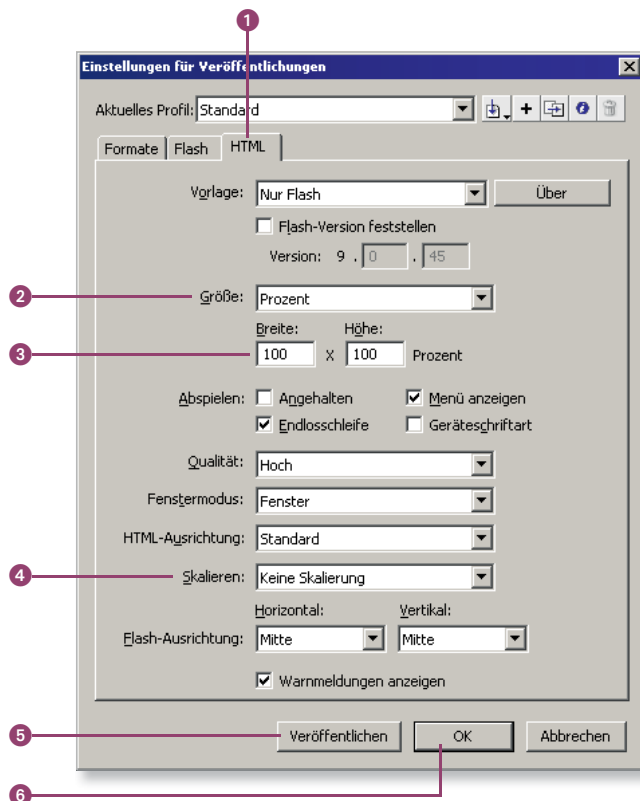


Abbildung 2 ►
Bis auf die Größe der Bühne sind die Dokumenteigenschaften in diesem Beispiel unwichtig.

Der Flash-Film soll das ganze Browserfenster ausfüllen und zentriert dargestellt werden. Um dies zu erreichen, öffnen Sie über DATEI • EINSTELLUNGEN FÜR VERÖFFENTLICHUNGEN... das entsprechende Fenster und ändern folgende Werte im HTML-Reiter ❶:

Die GRÖSSE ❷ wollen Sie in PROZENT angeben, wobei die BREITE und HÖHE jeweils 100 % ❸ betragen sollen. Der angezeigte Flash-Film soll zudem nicht skaliert, sondern in seiner ursprünglichen Größe angezeigt werden. Deshalb wählen Sie bei SKALIEREN den Eintrag KEINE SKALIERUNG ❹. Anschließend veröffentlichen Sie die Datei durch einen Klick auf VERÖFFENTLICHEN ❺ und bestätigen die Änderungen dann mit OK ❻.



Um Ihren eigenen Skype-Status anzeigen zu können, benötigen Sie außer Ihrem Skype-Benutzernamen nicht sehr viel. Sollten Sie keinen eigenen Skype-Account haben, können Sie den im Beispiel benutzten Accountnamen verwenden (flashfastforward).

Geben Sie einfach die folgende Zeile in die Adresszeile Ihres Webbrowsers ein:

<http://mystatus.skype.com/balloon/flashfastforward>

Schritt 4

Einstellungen für Veröffentlichungen anpassen

Bildrate

Wenn Sie keine Animationen in der Flash-Datei anlegen wollen, kann die Bildrate ohne Probleme bei den als Standard eingestellten 12 bps bleiben. Wenn Sie aber überlegen, Animationen einzubauen, sollten Sie die Bildrate auf mindestens 24 oder 25 bps hochsetzen – dies garantiert einen flüssigeren Ablauf des Flash-Films. Bei Spielen oder ähnlichen Anwendungen sollte die Bildrate noch weiter erhöht werden: Zwischen 30 und 50 bps sind in diesen Fällen keine Besonderheit.

◀ Abbildung 3

Einstellungen für Veröffentlichungen

Schritt 5

Skype-Status ermitteln



▲ Abbildung 4

Verschiedene Online-Status

Tipp

Sollte Ihnen die Art der Status-Darstellung nicht gefallen, können Sie auf der Skype-Website zwischen unterschiedlichen Darstellungsweisen wählen. Mehr dazu unter:
<http://www.skype.com/intl/de/share/buttons/advanced.html#status>

Schritt 6 Skype richtig konfigurieren

Der Browser lädt eine Grafik, die Ihren (bzw. unseren) Online-Status anzeigt.

Die Grafiken sind zum größten Teil selbsterklärend. Eine Besonderheit gibt es dennoch. Sollten Sie die Grafik »I'm not telling« (dt.: »Ich sag's nicht«) sehen, hat der Skype-User, dessen Online-Status Sie abfragen wollen, diese Funktion nicht freigegeben. Sollte das der Fall sein, erklärt Ihnen der nächste Schritt, wie Sie dieses Problem beheben. Wenn Sie eine der anderen Grafiken sehen, können Sie den nächsten Schritt überspringen und gleich mit Schritt 7 fortfahren.

Wenn Sie Skype installiert haben, die Abfrage Ihres Online-Status jedoch mit der Meldung »I'm not telling« fehlgeschlagen ist, werden Sie dies nun korrigieren.

Öffnen Sie hierzu das Skype-Hauptfenster, und gehen Sie über AKTIONEN • OPTIONEN... in den Optionen-Dialog. Hier wählen Sie den Menüpunkt PRIVATSPHÄRE ① und überprüfen, ob die Checkbox MEIN STATUS DARF IM NETZ VERÖFFENTLICHT WERDEN ② aktiviert ist. Ist das nicht der Fall, setzen Sie dort das Häkchen und bestätigen die Änderung mit SPEICHERN ③.

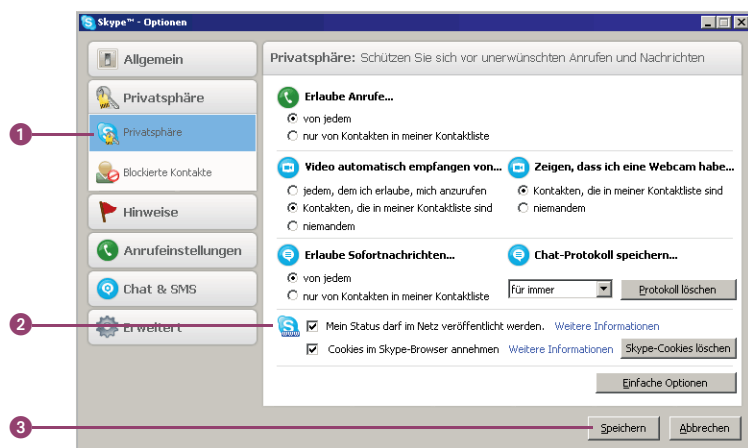


Abbildung 5 ►
Skype-Optionen einstellen

Falls Sie die angesprochene Option nicht unter dem Menüpunkt PRIVATSPHÄRE finden, müssen Sie sie über ERWEITERTE OPTIONEN ④ anzeigen.

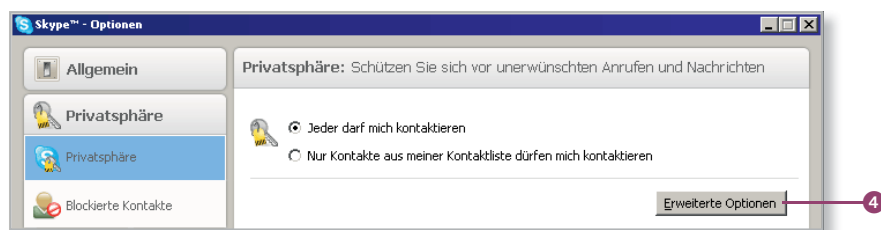


Abbildung 6 ►
Erweiterte Optionen anzeigen lassen

Wenn Sie nun Ihren Status über den Browser abfragen, sollte die entsprechende Grafik zu sehen sein.

Jetzt wollen Sie Ihren Status in Ihrem Flash-Film anzeigen. Um das zu erreichen, müssen Sie die Grafik vom Skype-Server `mystatus.skype.com` in den Flash-Film laden.

Hierzu benötigen Sie einen `Loader`, der über `URLRequest` das Bild lädt. Benennen Sie die bereits erstellte Ebene in »action« um, wählen Sie das erste Bild aus, und öffnen Sie mit `[F9]` bzw. am Mac mit `[Alt]+[F9]` das AKTIONEN-Fenster. Importieren Sie zuallererst die benötigten Klassen:

```
1: import flash.display.Loader;
2: import flash.net.URLRequest;
```

Jetzt können Sie auf die beiden Klassen zugreifen. Um das Bild zu laden und anschließend anzuzeigen, erzeugen Sie in den nächsten Zeilen den `Loader`, der das Bild per `URLRequest` vom Skype-Server lädt:

```
3: var mySkypeUsername:String = "flashfastforward";
4: var loader:Loader = new Loader();
5: loader.x = 10;
6: loader.y = 10;
7: loader.load(new URLRequest("http://mystatus.skype.com/
    balloon/"+mySkypeUsername));
```

In den Zeilen 5 und 6 wird der `Loader` um jeweils 10 Pixel auf der x- bzw. y-Achse verschoben. Ansonsten würde die Grafik zu weit oben links in der Ecke dargestellt. In Zeile 7 stoßen Sie den Ladevorgang des Bildes an.

Wenn Sie nun den Flash-Film veröffentlichen, werden Sie nichts sehen. Denn bevor der Status angezeigt werden kann, müssen Sie den `Loader` noch auf der Bühne platzieren. Ergänzen Sie folgende Zeile am Ende Ihres Skripts:

```
addChild(loader);
```

Testen Sie mit `[Strg]/[⌘]+[↵]` den Film. Der Skype-Status ist nun im Flash-Film zu sehen (siehe Abbildung 7).

Schritt 7

Skype-Status in Flash anzeigen



▲ **Abbildung 7**
Ihr aktueller Skype-Status wird im Flash-Film angezeigt.

Schritt 8

Flash-Film testen

Info

Da die Grafik direkt vom Skype-Server geladen wird, müssen Sie zum Zeitpunkt des Testens online sein.

Schritt 9

Einen eigenen Avatar einbauen

Avatar

Unter einem Avatar versteht man eine künstliche Person oder einen grafischen Stellvertreter einer echten Person in der virtuellen Welt. Avatare werden sehr häufig in Computerspielen benutzt.

Damit die Status-Grafik nicht ganz allein im Flash-Film zu sehen ist, setzen Sie einen Avatar (oder ein Foto) von sich darunter.

Erzeugen Sie als Nächstes eine Ebene »avatar« **1** unterhalb der Ebene »action«. Dann suchen Sie sich einen Avatar oder ein Foto aus und importieren das Bild über DATEI • IMPORTIEREN • IN BÜHNE IMPORTIEREN... Platzieren Sie das Bild so, dass es unterhalb der nachgeladenen Status-Grafik liegt **2**.

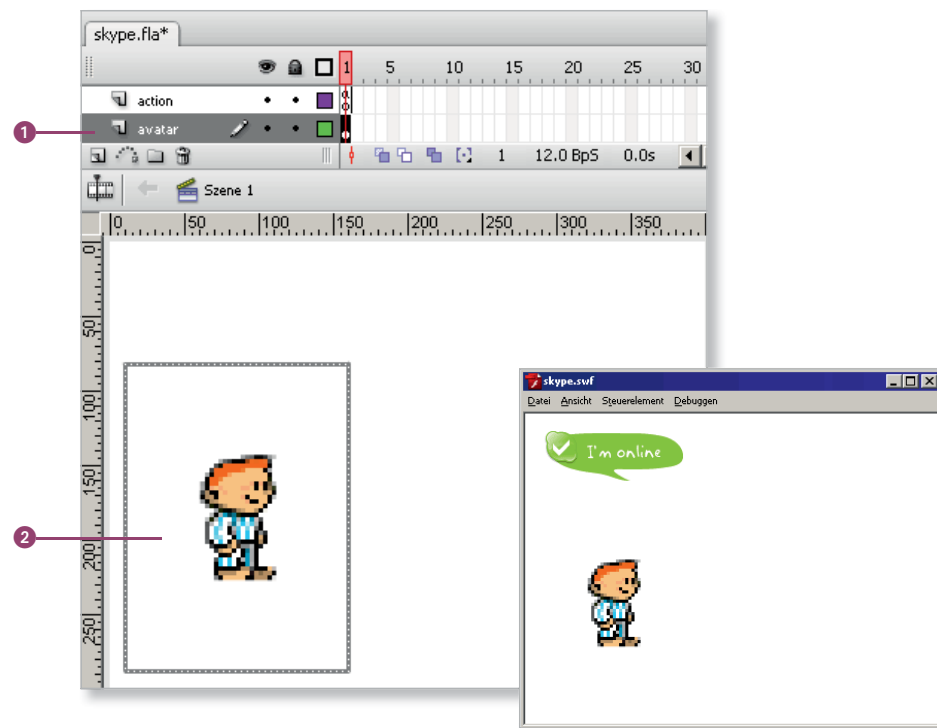


Abbildung 8 ►

Als Avatar kann eine einfache Comicfigur dienen.

Abbildung 9 ►►

Skype-Status mit Avatar

Schritt 10

Flash-Film testen

Wenn Sie den Flash-Film nun erneut über **Strg**/**⌘** + **↵** testen, sollte es so aussehen, als ob Ihr Avatar die Status-Nachricht »spricht«.

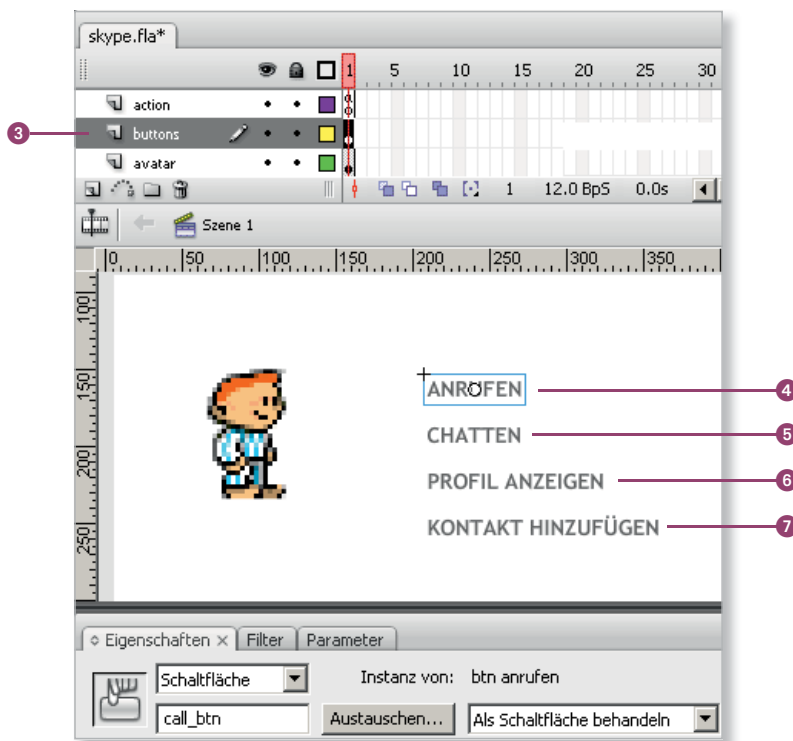
Schritt 11

Schaltflächen anlegen

Damit der Besucher auf Ihrer Seite nicht nur Ihren Online-Status sieht, sondern auch mit Ihnen kommunizieren kann, erweitern Sie das aktuelle Beispiel um einige Funktionen, die Skype zusätzlich anbietet.

Hierzu brauchen Sie vier Schaltflächen, die Sie anschließend mit den Funktionen verknüpfen.

Legen Sie oberhalb der Ebene »avatar« eine weitere Ebene »buttons« **3** an. Danach erstellen Sie die Schaltflächen wie beispielhaft im folgenden Bild gezeigt.



◀ **Abbildung 10**

Mit diesen vier Buttons können Besucher Ihrer Seite mit Ihnen in Kontakt treten.

Benennen Sie die Schaltflächen im Eigenschaftenfenster mit:

- ▶ »call_btn« 4
- ▶ »chat_btn« 5
- ▶ »profile_btn« 6
- ▶ »add_btn« 7

Um die Schaltflächen mit einer Funktion zu belegen, benötigen Sie nun eine weitere Flash-interne Klasse: `MouseEvent`. Importieren Sie diese wie folgt direkt unter den anderen Import-Anweisungen:

```
import flash.events.MouseEvent;
```

Jetzt fügen Sie jeder Schaltfläche einen EventListener am Ende Ihres Skripts hinzu, der auf `MouseEvent.CLICK` »lauscht«:

```
call_btn.addEventListener(MouseEvent.CLICK, onClick);
chat_btn.addEventListener(MouseEvent.CLICK, onClick);
profile_btn.addEventListener(MouseEvent.CLICK, onClick);
add_btn.addEventListener(MouseEvent.CLICK, onClick);
```

Schritt 12

Funktionen für die Schaltflächen implementieren

Alle Schaltflächen rufen bei einem Mausklick die Funktion `onClick()` auf. Diese implementieren Sie wieder am Ende Ihres Skripts unter den eben eingefügten Codezeilen:

```
1: function onClick(evt:MouseEvent):void
2: {
3:     var command:String;
4:     switch(evt.target)
5:     {
6:         case call_btn: command = "call"; break;
7:         case chat_btn: command = "chat"; break;
8:         case profile_btn: command = "userinfo"; break;
9:         case add_btn: command = "add"; break;
10:    }
11:    navigateToURL(new URLRequest("skype:"+mySkypeUsername+"?"
    +command), "_self");
12: }
```

Der Funktion wird in Zeile 1 das gefeuerte `MouseEvent` als Parameter mit übergeben. Je nachdem, welche Schaltfläche das Ereignis ausgelöst hat (Zeile 4), wird dem `command`-String ein anderer Wert zugewiesen (Zeilen 6 bis 9).

Anschließend wird mit Hilfe der `navigateToURL`-Funktion in Zeile 11 die entsprechende URL im aktuellen Browserfenster aufgerufen.

`chat_btn` z.B. ruft folgende Seite auf:

`skype:flashfastforward?chat`

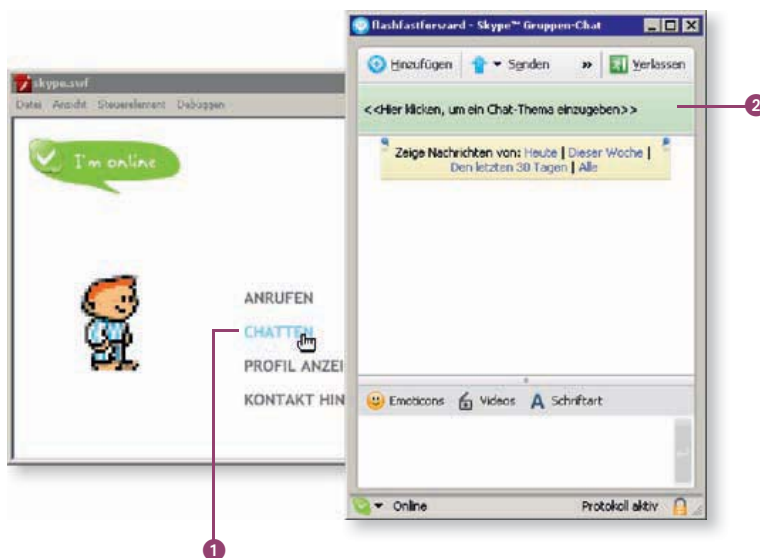
Wie Sie sicherlich schon erkannt haben, ist das keine normale `http`-Adresse, sondern eine `skype`-Adresse. So teilen Sie dem Browser mit, dass nicht er die Seite öffnen soll, sondern dass er die Anfrage an das Skype-Programm weitergeben soll.

Skype erhält damit den Namen des Users `flashfastforward` und die Information, wie dieser weiterverarbeitet werden soll: `chat`. Das Programm öffnet in diesem Fall ein Chat-Fenster mit dem Benutzer `flashfastforward`.

Schritt 13

Flash-Film testen

Wenn Sie nun den Flash-Film über **STEUERUNG • FILM TESTEN** ansehen und auf **CHATTEN** klicken ❶, sollte Skype ein Chat-Fenster öffnen ❷.



◀ **Abbildung 11**

Ein Klick auf den Button CHATTEN öffnet Skype auf dem Rechner des Users.

Es kann natürlich passieren, dass ein Besucher Ihrer Seite kein Skype installiert hat. Dann funktionieren die speziellen Skype-Aufrufe nicht und der Besucher wundert sich, was die Schaltflächen überhaupt bewirken. Um diesen Fall abzufangen, können Sie bei jedem Besucher überprüfen, ob er das Skype-Programm auch installiert hat.

Die Skype-Entwickler haben ein JavaScript geschrieben, das diese Kontrolle für Sie übernimmt. Binden Sie die JavaScript-Datei einfach in Ihre HTML-Seite ein. Dazu fügen Sie lediglich eine Zeile in den `<head>`-Bereich Ihrer SKYPE.HTML-Datei ein:

```
<script type="text/javascript" src="http://download.skype.com/
share/skypebuttons/js/skypeCheck.js"></script>
```

Um dieses JavaScript aus dem Flash-Film heraus anzusprechen, nutzen Sie die Klasse `ExternalInterface`, die Ihnen Flash zur Verfügung stellt. Auch diese müssen Sie am Anfang Ihres Skripts importieren. Ergänzen Sie unter die letzte Import-Anweisung folgende Codezeile:

```
import flash.external.ExternalInterface;
```

Nachdem Sie die Klasse importiert haben, verändern Sie den restlichen Code so, dass die Schaltflächen nur mit Funktionen belegt werden, wenn Skype installiert ist. Die vier EventListener aus Schritt 12 werden in eine `if-else`-Anweisung verpackt:

Schritt 14

Ist Skype auch installiert?

Schritt 15

Skype-Check per ExternalInterface

```

1: if(ExternalInterface.available && !Boolean(
    ExternalInterface.call("skypeCheck")))
2: {
3:     call_btn.enabled = false;
4:     chat_btn.enabled = false;
5:     profile_btn.enabled = false;
6:     add_btn.enabled = false;
7: }
8: else
9: {
10:    call_btn.addEventListener(MouseEvent.CLICK, onClick);
11:    chat_btn.addEventListener(MouseEvent.CLICK, onClick);
12:    profile_btn.addEventListener(MouseEvent.CLICK, onClick);
13:    add_btn.addEventListener(MouseEvent.CLICK, onClick);
14: }

```

Die `if`-Anweisung kontrolliert zum einen, ob das `ExternalInterface` überhaupt zu erreichen ist; zum anderen wird die JavaScript-Funktion `skypeCheck` aufgerufen, die die eigentliche Installations-Überprüfung durchführt. Sollte `skypeCheck` zurückliefern, dass Skype nicht installiert ist, werden die Schaltflächen in den Zeilen 3 bis 6 deaktiviert. Ist die Überprüfung erfolgreich, fügen Sie in den Zeilen 10 bis 13 den Schaltflächen die benötigten `EventListener` hinzu.

Schritt 16 Der letzte Feinschliff

Um dem Ganzen nun noch den finalen Schliff zu verpassen, erstellen Sie eine weitere Ebene »graphics« ¹. Hier geben Sie dem »Menü« den Titel »Skype-Optionen« und zeichnen einen Rahmen darum ².

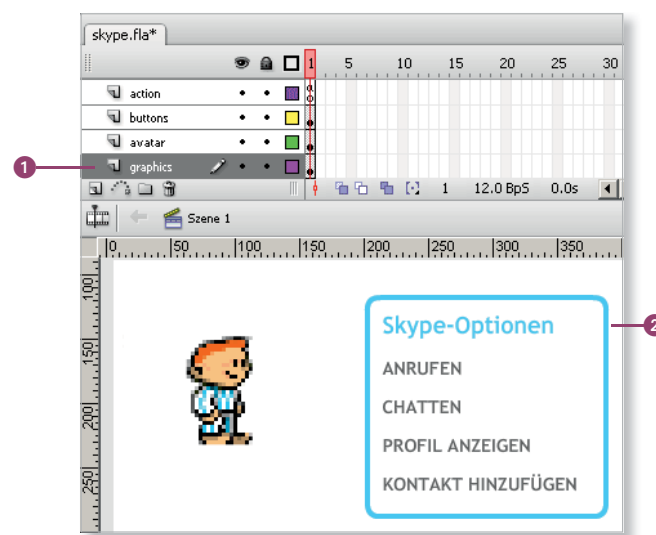
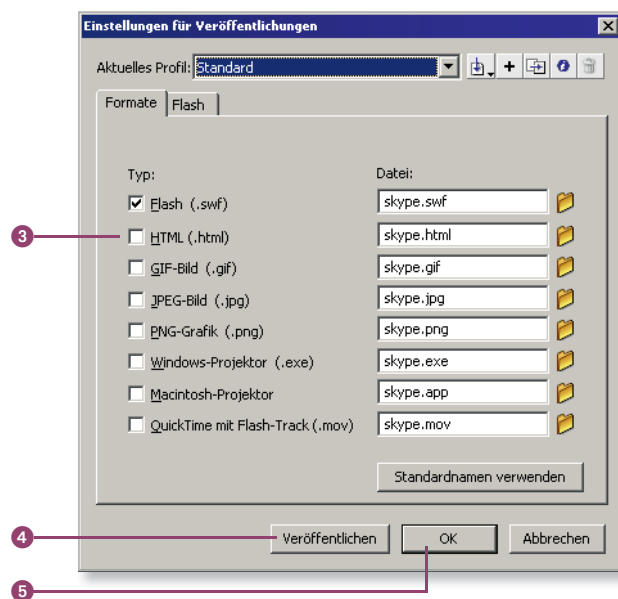


Abbildung 12 ►

Durch den Titel und einen Rahmen werden die Buttons besser hervorgehoben und sind auf Anhieb verständlich.

Weil Sie Ihren Flash-Film in der Flash-Autorenumgebung nicht innerhalb der html-Datei aufrufen, sondern im Flash Player, fehlt beim Testen in Flash die Funktionalität der ExternalInterface-Klasse. Bevor Sie Ihren Film veröffentlichen, müssen Sie noch in den EINSTELLUNGEN FÜR VERÖFFENTLICHUNGEN – die Sie über **Strg/Alt+U+F12** erreichen – die HTML-Option **3** deaktivieren. Sonst überschreiben Sie Ihre gerade geänderte html-Datei wieder.

Veröffentlichen Sie dann den Flash-Film durch einen Klick auf VERÖFFENTLICHEN **4**, und bestätigen Sie anschließend alles mit OK **5**.



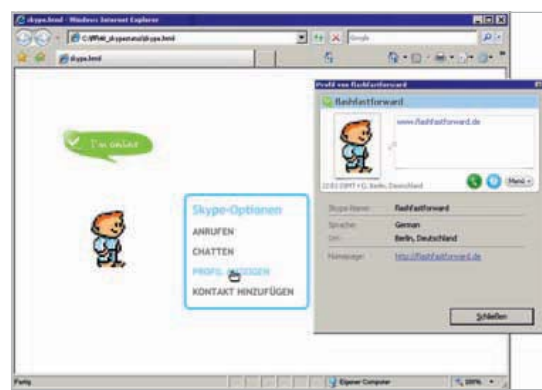
Schritt 17

Einstellungen für Veröffentlichungen ändern

Abbildung 13

Da Sie die html-Datei angepasst haben, dürfen Sie nur eine swf-Datei veröffentlichen.

Jetzt öffnen Sie die Datei SKYPE.HTML in einem Webbrowser Ihrer Wahl. Wenn Sie Skype nicht installiert haben, funktionieren die Schaltflächen nicht. Anderenfalls sollten Sie bei einem Klick auf z.B. PROFIL ANZEIGEN Folgendes sehen:



Schritt 18

Flash-Film im Browser testen

Abbildung 14

Profil anzeigen

Die DVD zum Buch

Auf der beiliegenden DVD zum Buch finden Sie drei Ordner mit den Namen **BEISPIELMATERIAL**, **TESTVERSION** und **VIDEO-LEKTIONEN**. Dort finden Sie folgende Inhalte:

Beispielmaterial

In diesem Ordner liegt das komplette Beispielmaterial für die Workshops. In der Regel sind das eine fla-Datei und die exportierte swf-Datei. Manche Workshops haben zusätzlich noch Bilder, eigene Klassen (as-Dateien) und Zwischenschritte, damit Sie, falls etwas einmal nicht auf Anhieb funktionieren sollte, den Zwischenstand prüfen können. Den Code können Sie direkt aus der fla- bzw. der swf-Datei kopieren, falls Sie nicht selber tippen möchten.

Testversion

Falls Sie Adobe Flash CS3 noch nicht auf Ihrem Rechner installiert haben, können Sie die 30-Tage-Testversion für PC bzw. Mac nutzen. Kopieren Sie einfach die entsprechende Datei auf Ihre Festplatte und leiten Sie durch einen Doppelklick den Installationsprozess ein.

Video-Lektionen

Als Ergänzung zum Buch möchten wir Ihnen ausgewählte Lehrfilme aus dem Video-Training »ActionScript 3« von Tobias Hauser und Armin Kappeler (ISBN 978-3-8362-1181-9) zeigen. Um das Training zu starten, gehen Sie auf der Buch-DVD in den Ordner **VIDEO-LEKTIONEN** und klicken dort auf der obersten Ebene die Datei **START-PC.EXE** (Windows) oder **START-MAC** (Mac). Alle anderen Dateien können Sie ignorieren.

Inhalt des Trainings:

Bitte klicken Sie im rechten Bereich auf einen Lektionsnamen und die Video-Lektion startet. Sie finden folgende Filme:

Kapitel 1: Einstieg in ActionScript 3

1.1 Neuerungen in ActionScript 3

1.2 Das neue Ereignismodell

Kapitel 2: Spracheinführung

2.1 Funktionen

2.2 Strings

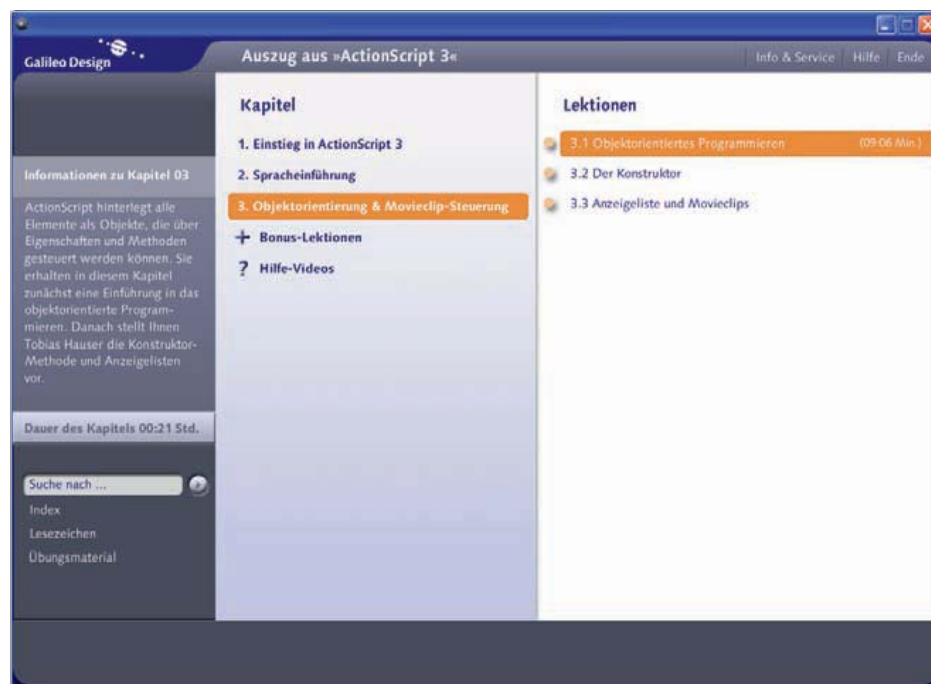
2.3 Arrays

Kapitel 3: Objektorientierung & Movieclip-Steuerung

3.1 Objektorientiertes Programmieren – Grundlagen

3.2 Der Konstruktor

3.3 Anzeigeliste und Movieclips



◀ **Abbildung 1**
Die Benutzeroberfläche des Video-Trainings

Index

3D

- Aussichtspunkt 205
- Brennweite 205
- Elemente in einem Kreis positionieren 223
- Exkurs 184
- Fluchtpunkt 185
- Kamerafahrt 198
- Koordinatensystem 184
- Performance optimieren 197
- Tiefenverwaltung 186
- Z-Sortierung 209, 225
- 3D-Bildergalerie 198
- 3D-Engine 188
 - programmieren 208
- 3D-Karussellmenü 218
- 3D-Positionsvariable 222
- 3D-Programmierung 184, 188
- 3D-Raum
 - Drehung 226
 - Elemente nach vorn holen 218
 - Maussteuerung 218, 226
 - Tiefe 225
- 3D-Sternenfeuer 188
- 3D-Umgebung 222
- _root 243
- _xscale
 - animieren 250
- _yscale
 - animieren 250

A

- Abgerundetes Rechteck
 - per Action-Script zeichnen 221
- Abprallen 168
- ActionScript 3
 - Loslassen der Maustaste 363
- ActionScript-Datei
 - erstellen 25
- Adobe AIR 380
- Adobe AIR 1.0 381
- Adobe-Lifedocs 292
- AIR 377
 - Einstellungen 381
 - geöffnete Datei verarbeiten 385
 - Window style 381
- AIR-Anwendung
 - Drag-Funktion 384

- Schließen-Funktion 385
- AIR-Musikplayer 380
- Alphakanal-Überblendung 292
- Alphaverlauf
 - hinzufügen 147
- Alphawert
 - auslesen 73, 116, 139
- AMFPHP 318
- Anfangsbeschleunigung 166
- Animation
 - loopen 36
 - neu anstoßen 36
- Anzeigeliste 105, 316
 - Exkurs 104
- Array
 - in einer for-Schleife durchlaufen 273
- attachMovie
 - Exkurs 104
 - Tiefenverwaltung 104
- Audio
 - Kanal auslesen 374
- Audio-Player 342
- Ausrichtung
 - im Browser 415
- Automatische Plug-in-Erkennung 408
- Avatar 458

B

- Ball
 - mit physikalischen Eigenschaften 163
 - Wurfkraft errechnen 172
- Ballsimulation 163
- Beschleunigungs-Methoden 35
- BevelFilter 130
- Bewegung
 - mit ActionScript nachzeichnen 17
 - nachfedern 179
 - nachschwingen 179
 - organisch animieren 112
 - weichzeichnen 49, 50
 - Zielposition bestimmen 33
 - zu einem Zufallspunkt 34
- Bewegungs-Tween
 - erstellen 65
- Bezeichner 64
- Bild
 - dynamisch laden 237
 - im dreidimensionalen Raum 198

- importieren 88, 97
- kopieren 92
 - mit ActionScript platzieren 53
- Partikelanzahl auslesen 114
- per Doppelklick ein- und auszoomen 245
- spiegeln 142, 146
- Bilddaten kopieren 85, 94
- Bildpartikel animieren 107
- Bildrate
 - bei Animationen 455
 - bei Spielen 455
 - wählen 455
- Bildteil
 - vergrößern 87
- Bildüberblendung 290
- Bitmap
 - mit ActionScript vergrößern 93
- BitmapData-Klasse 76
- BitmapData-Objekt 76, 100
 - anzeigen 77
- Bitmapdaten
 - kopieren 79
 - vergrößern 93
- BlurFilter 136
- BlurFilter-Objekt 49
- Bogenmaß 148
- Boolesche Variable 170
- Brennweite 185
- Browser
 - Vor- und Zurück-Button 433
- Browsernavigation
 - in Flash abfangen 433
- Bühnenrand
 - Kollision überprüfen 158
- Buttonaktion
 - programmieren 394
- Buttonklick
 - abfragen 393
- buttonMode 446

C

- cacheAsBitmap 259
- ClipLoader 234
- Container-Movieclip
 - anlegen 61
- ContextMenu 270

D

- Dateibrowser 266
- Dateityp
 - prüfen 386
- Dateiupload 263
 - PHP-Skript 266
- Daten
 - auf einen Server übertragen 263
- Datenbank 319
- Deeplink 425, 433
- Diashow 290
- DIV-Container 409
- Document Object Model 301
- Dokumenteigenschaften 176
 - anpassen 30
- Dokumentklasse 109
 - Definition 190
 - verknüpfen 110
- DOM 301
- Doppelklick
 - mit ActionScript 2 245
 - testen 249
- Doppelklick-Klasse 245
- Drag & Drop 47
 - implementieren 363
- Drehung
 - um die y-Achse 226
- Dreidimensionaler Effekt 184
- Dreidimensionaler Raum
 - Position angeben 206
- DropShadowFilter 173
- Durchsuchen-Button 266
- Dynamisches Textfeld 25
 - Variable aus HTML 429

E

- Ease-Klasse 111
- Ease-Methode 35, 112
- Effekt
 - Feuer 131
 - Flüssigkeit 120
 - Lupe 87
 - Partikel 107
 - Schreibmaschine 24
 - Schrift 17
 - Scrollen 42
 - TV-Rauschen 75
 - Überblendung 292
 - Verwischen 79
 - Wasser 120

- Weichzeichnung 42
- Wellen 51
- Eigenschaft
 - physikalische 163
- Einstellungen für Veröffentlichungen 407
- ENTER_FRAME-Event 49
- EOLAS-Patent 408
- Event
 - senden 240
- EventDispatcher 240
- EventListener 36
- Exporteinstellungen
 - AIR 381
- Export für ActionScript 64
- Externe Bilder
 - laden 290
- Externe Datei
 - Ladefehler 240
 - Loader-Klasse 234
- Externe Grafik
 - in Flash-Film laden 457

F

- Fallbewegung 157
 - mit der Maus steuern 160
- Fallgeschwindigkeit 157
- Farbverlauf
 - Farben festlegen 133
 - linear 133
 - radial 122
- Farbwähler 96
- Farbwert
 - als Zeichenkette ausgeben 102
 - auslesen 96, 116
 - bei Mausbewegung auslesen 102
 - ersetzen 139
- Fehlermeldung
 - ausgeben 240
- Fernseher-Rauschen
 - simulieren 75
- Feuerbewegung
 - simulieren 135
- Feuer-Effekt 131
- FileReference-Klasse 265
- Filter 87
- Flammen
 - erzeugen 131, 136, 139
- Flash-Film

- mit SWFObject einbinden 450
- Flash-Filter 42
- Flash-Formular 311
- Flash-Navigation 433
- Flash Player
 - Test mit unterschiedlichen Versionen 421
 - Update 418
 - Version 410
 - Versionscheck 418
- Flash Player-Version 420
- Flash Player-Versionserkennung 418
- Flash-Plug-in
 - automatisch aktualisieren 422
- Flash Remoting 311
 - Exkurs 309
- Flash-Video 387
- Flex 377
- Flüssigkeit simulieren 120
- flv-Format 387
- Formular
 - Fehlermeldung 338
 - Tab-Reihenfolge 315
- Formulareingaben überprüfen 336
- Führungsebene 18

G

- Gästebuch 311
 - Einträge auslesen 321
 - Einträge erstellen 333
 - Erstellungszeitpunkt 330
 - Formular 315
- Gateway-Konfiguration 324
- Geometrische Form
 - zeichnen 176
- Getter-Methoden 113
- Glow-Filter 87, 181, 197
 - erstellen 92
- gotoPage (Funktion) 448

H

- HTML
 - Seitentitel 451
 - Variable übergeben 425

I

- ID3-Eigenschaft 353
- ID3-Tag
 - auslesen 351
- Info-Palette 43

In HTML einbinden 406
Instantiierung 13
Instanzname vergeben 44
In Symbol konvertieren
 Dialogfenster 31
Internet Explorer 407, 408

K

Kabel
 mit der Maus bewegen 178
Kabelschwingung
 simulieren 175
Kamerafahrt 198, 213
Klasse 12
 anlegen 26
 importieren 62
 Variablen deklarieren 26
 vererben 32
Klassenpfad
 angeben 436
Klassen-Variable 25, 34
Klasse und Objekt 13
Klickhand 231
Kommunikation
 mit HTML 425
Komponente 311, 313
Komponenten-Fenster 264
Konstruktor 26
Kontextmenü
 anpassen 268
 eigener Menüeintrag 272
 individuelles 268
 Standard-Menüeintrag
 ausblenden 270
 Trennlinie darstellen 274
Kontextmenü-Klasse 268
Kreis
 mit ActionScript zeichnen 90
Kurve
 Linieigenschaften festlegen 177
 mit ActionScript zeichnen 176
 Zielkoordinaten 177

L

Ladebalken
 erzeugen 242
Ladefehler
 berücksichtigen 240
Ladefortschritt

 anzeigen 243
Ladestatus
 anzeigen 240
 errechnen 241
Landingpage 418, 423
Laufband 279
 Breite bestimmen 286
 Geschwindigkeit 288
 Maske erstellen 286
Lautstärke 361
 Amplitude 367
 Kanal 366
 visualisieren 366
Linie
 mit ActionScript zeichnen 41, 59
Listener 242
Loader-Klasse
 für ActionScript 2 234
Lupe 87
 mit der Maus verfolgen 95

M

Map 126
 animieren 136
Maske
 animieren 51, 64, 65
 dynamisch erstellen 286
 erstellen 43
 in Movieclip konvertieren 43
 runde 87
 zuweisen 56
Maskenebene 65
 animieren 67
Masken-Tween 64
Math.atan2 37
Math.random 78, 113
Math.round 33
Matrix-Objekt 93, 146
Maus
 Pflanzen wachsen lassen 58
 Werkzeugspitze erstellen 126
Mausbewegung
 Lupe erstellen 94
 nachzeichnen 58
 verfolgen 178
Maus-Event
 Movieclip erzeugen 191
Maus-Listener
 entfernen 402
Mausposition
 aktuelle 60
Mauszeiger
 ändern 230
 Klickhand 231
Menü
 über XML steuern 302
Menüpunkt
 aus XML erstellen 305
 Klick abfangen 447
 mit Funktionalität ausstatten 306
 Rollover 445
Methode
 private 249
MouseEvent 116
 hinzufügen 170
MouseEvent-Klasse 447
MouseListener 192
Movieclip
 auf einem Hintergrund platzieren
 98
 dynamisch erzeugen 155
 maskieren 48
 Registrierungspunkt 108
 unsichtbar schalten 124
MovieClip-Klasse
 Zeichnen-Methoden 176
MovieClipLoader 236
MP3-Player 342
 Display 344
 Endlosschleife 355
 Hintergrundgrafik 343
 ID3-Daten 352
 Ladestatus anzeigen 240
 Lautstärke regulieren 361
 Musikstück hinzufügen 385
 Spektrumsanzeige 370
 Statusbalken 345
 Steuerungselement 346
MySQL-Datenbank 319

N

navigateToURL 460
Navigation
 Menü-Element 443
 Seitenwechsel 448
 Sprungmarke 441
 Unterseiten erstellen 440
noise-Methode 77

null 335

O

Objektorientierte Programmierung 12
onLoadError 240
onLoadProgress 241
onMouseDown 60
onMouseMove 94
onMouseUp 60
Optische Täuschung 51

P

Package 12
paletteMap 140
Partikel
 Anzahl auslesen 114
 Bewegungsskript 117
 erstellen 115
 Richtung der Animation 113
Partikelanimation 107
Partikel-Effekt 107
Performance 259
Perspektive 184
Pfadanimation 17
Pfad hinzufügen 18
Pflanzengenerator 58
Physikalische Eigenschaft 166, 175, 179
Plastischer machen 130
Playlist 342
Plug-in-Check 418
Plug-in-Switcher 421
Point-Klasse 62, 177
Positionierung
 zufallsgesteuerte 33
Prozedurale Programmierung 12

Q

Qualitätseinstellungen 417

R

Radialer Farbverlauf 122
Räumlichkeit
 simulieren 156
Rauschen 75
 animieren 78
Rechteck
 mit ActionScript erzeugen 220
 zufällige Füllfarbe 221
Reflexion 141
Registrierungspunkt 108

Remoting-Gateway
 installieren 318
Remoting-Service 320
Rich Internet Applications 377
Rollover-Menü 443
root-Knoten 300
Rotation 37
Rotationsmatrix 146
Rotations-Tween 38
Rückgabewert 102

S

Schatten 173
Schlagschatten 260
Schneefall simulieren 152
Schreibmaschinen-Effekt 24
Schrift-Effekt 17
Schriftzug nachzeichnen 17
Schwingung 175
 animieren 179
Scrollbalken anlegen 45
Scroll-Effekt 42
Scrollen
 weichzeichnen 42
Setter-Methoden 113
Sicherheitseinstellungen 294
Sichtbarkeit
 von Movieclips 125
Simulation
 Schneefall 152
Sinusfunktion 159
Sinuskurve 160
Skalierung im Browser 414
Skalierungsmatrix 93
Skype
 ansprechen 460
 aus Flash ansprechen 460
 Avatar 458
 konfigurieren 456
Skype-Status
 ermitteln 453, 455
 in Flash anzeigen 457
Slideshow 290
 Bildwechsel 298
 Intervalldauer 298
 starten 297
Soundsteuerung 357
Soundstream
 auslesen 374

Spektrum
 animieren 375
 rechter und linker Kanal 374
Spektrumsanzeige 370
Spiegelung
 mit Alphaverlauf 141
Spinne
 animieren 29
 Beine animieren 39
 Netz spinnen 40
Springende Bälle 163
startDrag 47
Statische Methode 247
Sternenleuchten 197
stopDrag 47
String
 in ein Array zerlegen 27
SWFAddress
 integrieren 448
SWFAddress-Skript 433
swf-Datei
 Hintergrundfarbe 410
SWFObject 406, 418, 425
 einbinden 409
 laden 408

T

Tab-Reihenfolge 315
Text
 Ausgabe per ActionScript 26
 einzelne Buchstaben ausgeben 27
 in einzelne Buchstaben zerlegen 27
 in Textfeld ausgeben 100
 Schriftart 25
 Schriftgröße 25
Textfeld
 dynamisch 25
threshold-Funktion 120
Tiefe
 simulieren 156
Tiefenverwaltung 104
 in ActionScript 3 106
Tooltipp 253
 zeichnen 257
trace 240
trackAsMenu 256
Transparenz 416
TV-Rauschen 75
Tweening

per ActionScript 255
Tween-Klasse 29, 255

U

Überblendung 292
URL
aufrufen 460
URLRequest 457
User Interface Komponenten 264

V

Variable
an Flash-Film übergeben 429
aus HTML 429
entgegennehmen 425
übergeben 425
Verdunstungs-Effekt 129
Vererbung 12, 14
Vergrößerungs-Effekt 87
Verknüpfungsname 283
Verlauf
Alphawerte definieren 148
Veröffentlichen 407
Ausrichtung 415
Browserspezifikation 406
Größe des Flash-Films 411
High-Quality 411
JavaScript-Datei 407
mit Flash 8 407
Probleme 407
Skalierung 414
transparent 411
Umleitung einbauen 423
zentriert 411
Verwischen-Effekt 79
Video

abspielen 391
Abspielposition anzeigen 396
Abspielzeit anzeigen 398
Endlosschleife 399
Format bestimmen 392
Ladefortschritt anzeigen 397
Lautstärke 400
Lautstärkereglern 400
spiegeln 149
Video-Player 387
VideoPlayer-Klasse 391
Vom Boden abprallen 168
Vom Bühnenrand abprallen 168
Vor- und Zurück-Button 395

W

Wassereffekt 120
Web 2.0-Reflexion 141
Website
Seitensteuerung 448
Seitentitel 451
Weichgezeichneter Scroller 42
Weichzeichnung 130
Weichzeichnungsfilter 49
Wellen-Effekt 51
Werfen 170
Werkzeugspitze
erstellen 122, 126
Winkelberechnung 37
Winkelverhältnis
berechnen 183
wmode 416

X

XML
Exkurs 300

mit ActionScript 3 301
XML-Datei
laden 304
Textknoten ignorieren 304
XML-Daten
Aufbau 300
auslesen 305
XML-Klasse 301
XML-Knoten 300
XML-Menü 302

Z

Zeichenkette 101
Zeichenprogramm
erstellen 60
Zeichnen
dynamisch 175
mit ActionScript 59
organisch 62
Zeichnen-Methode 175
Zeichnen mit ActionScript
abgerundetes Rechteck 221
Außenlinie 181
Blätter 68
Blüten 65
Kreisfunktion 90
Kurve 177
weiche Linien 63
Zoomeffekt 250
per Doppelklick 245
Z-Sorting 186
Zufälliger Farbwert 221
Zufallsanimation 29
Zufallsgesteuerte Positionierung 33
Zufallsposition 155
Zufallszahl 78, 113

Bibliografische Information der Deutschen Bibliothek

Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.ddb.de> abrufbar.

ISBN 978-3-8362-1065-2

© Galileo Press GmbH, Bonn 2008

1. Auflage 2008

Der Name Galileo Press geht auf den italienischen Mathematiker und Philosophen Galileo Galilei (1564–1642) zurück. Er gilt als Gründungsfigur der neuzeitlichen Wissenschaft und wurde berühmt als Verfechter des modernen, heliozentrischen Weltbilds. Legendär ist sein Ausspruch *Eppur se muove* (Und sie bewegt sich doch). Das Emblem von Galileo Press ist der Jupiter, umkreist von den vier Galileischen Monden. Galilei entdeckte die nach ihm benannten Monde 1610.

Lektorat Katharina Geißler

Korrektorat Heike-Maria Schmidt, Oberursel

Herstellung Vera Brauner

Einbandgestaltung Hannes Fuß, www.exclam.de

Typografie und Layout Vera Brauner

Satz Markus Miller, München

Druck Himmer AG, Augsburg

Dieses Buch wurde gesetzt aus der Linotype Syntax (9,25 pt/13 pt) in Adobe InDesign CS3. Gedruckt wurde es auf Bilderdruckpapier (115 g/m²).

Gerne stehen wir Ihnen mit Rat und Tat zur Seite:

katharina.geissler@galileo-press.de

bei Anmerkungen zum Inhalt des Buches

service@galileo-press.de

für versandkostenfreie Bestellungen und Reklamationen

ralf.kaulisch@galileo-press.de

für Rezensionen- und Schulungsexemplare

Das vorliegende Werk ist in all seinen Teilen urheberrechtlich geschützt. Alle Rechte vorbehalten, insbesondere das Recht der Übersetzung, des Vortrags, der Reproduktion, der Vervielfältigung auf fotomechanischem oder anderen Wegen und der Speicherung in elektronischen Medien.

Für die Beispieldateien auf dem Datenträger: Sönke Kluth, Philipp Kyeck, Frederik Schrick, Aron Woost und Lizenzgeber (www.fotolia.de). Alle Rechte vorbehalten. Alle auf dem Datenträger zur Verfügung gestellten Beispielmateriale sind ausschließlich zu Übungszwecken bestimmt. Jegliche weitere Verwendung bedarf der schriftlichen Genehmigung des Urhebers.

Ungeachtet der Sorgfalt, die auf die Erstellung von Text, Abbildungen und Programmen verwendet wurde, können weder Verlag noch Autor, Herausgeber oder Übersetzer für mögliche Fehler und deren Folgen eine juristische Verantwortung oder irgendeine Haftung übernehmen.

Die in diesem Werk wiedergegebenen Gebrauchsnamen, Handelsnamen, Warenbezeichnungen usw. können auch ohne besondere Kennzeichnung Marken sein und als solche den gesetzlichen Bestimmungen unterliegen.