Sönke Kluth, Frederik Schricker,
Philipp Kyeck, Aron Woost

Flash CS4

Die Workshops für Einsteiger und Fortgeschrittene

Galileo Press

Zufallsanimation

Mit Hilfe der TweenLite-Engine krabbelt eine Spinne zufallsgesteuert hin und her

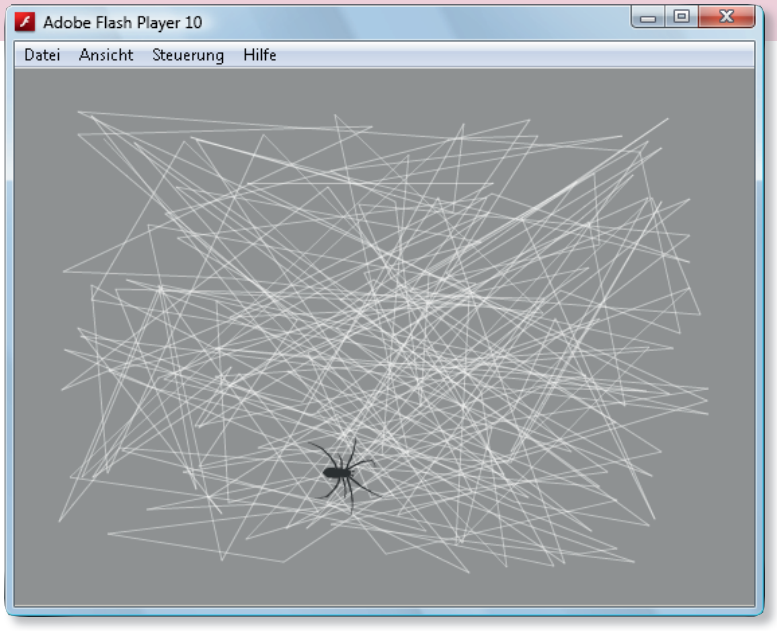
Anstatt wie bisher Movieclips aufwendig über die Zeitleiste zu animieren, sollten Sie zukünftig darüber nachdenken, ob Sie die gleiche Animation nicht einfacher per Skript erledigen könnten. In diesem Workshop lernen Sie den Umgang mit der TweenLite-Engine kennen. Dabei kommt es weniger auf eine naturgetreue Darstellung einer Spinnenbewegung an als auf das Zeigen der TweenLite-Funktionen.

Ziel

Eine zufällig gesteuerte Animation erstellen

- TweenLite-Engine benutzen
- EventListener implementieren
- Animationsloop erstellen

[**Ergebnis:** *02_Zufallsanimation/*.fla*]



Vorbereitungen 1

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.

Erstellen der Flash-Datei 2

Wechseln Sie zu Flash und erstellen Sie über Strg/⌘+N eine neue FLASH-DATEI (ACTIONSCRIPT 3.0) ❶ – bestätigen Sie den Dialog mit OK ❷.

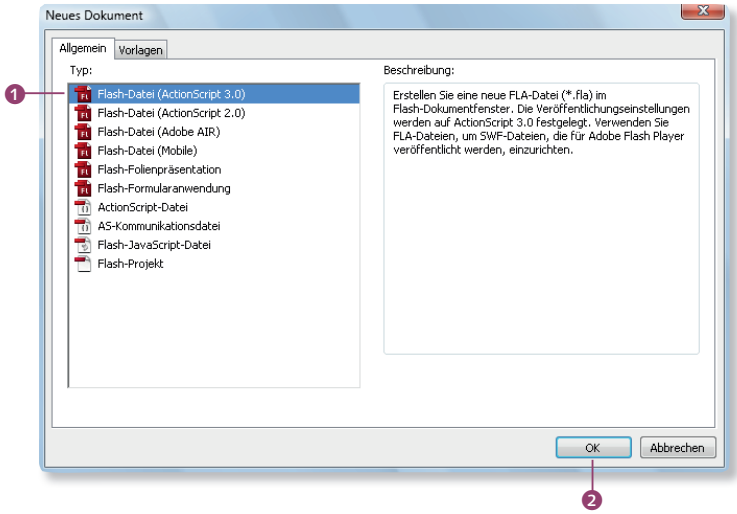


Abbildung 1 ▸
Legen Sie ein neues Flash-Dokument an.

Diese Datei speichern Sie über das Menü DATEI • SPEICHERN als »Spinne.fla« in dem in Schritt 1 erstellten Ordner ab.

Anpassen der Dokumenteigenschaften 3

Anschließend passen Sie die DOKUMENTEIGENSCHAFTEN über Strg/⌘+J an: Die GRÖSSE belassen Sie bei den eingestellten 550×400 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 ❻.

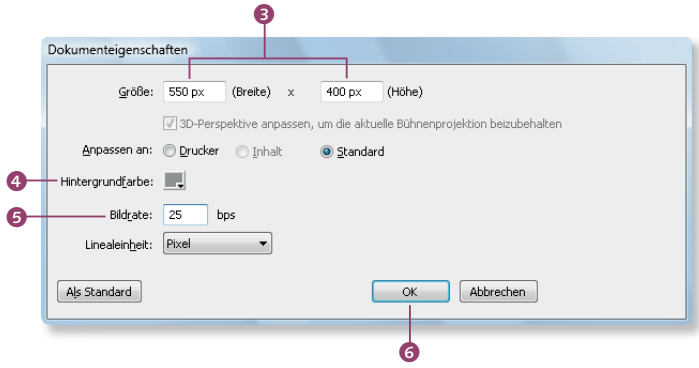
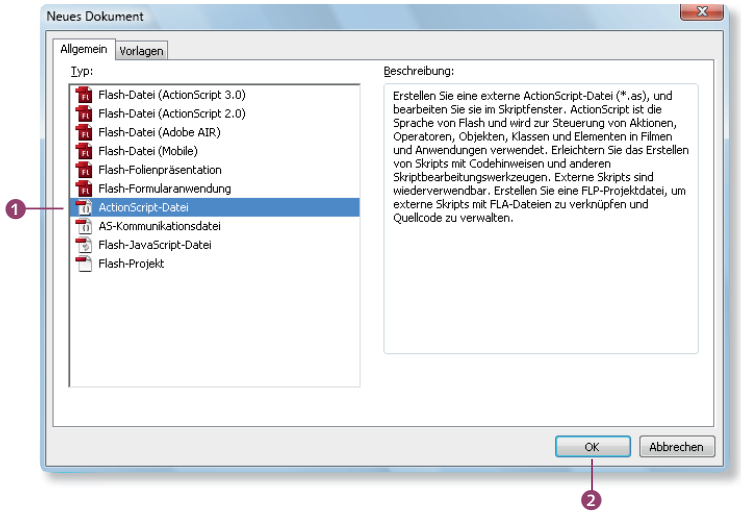


Abbildung 2 ▸
Die Hintergrundfarbe ändern Sie durch einen Klick auf das graue Kästchen neben HINTERGRUNDFARBE. Anschließend wählen Sie mit der Pipette eine neue Farbe.

4 Anlegen der Spider-Klasse

In diesem Schritt werden Sie eine Basisklasse erstellen, die der Spinne später Leben einhauchen wird. Zu diesem Zweck öffnen Sie mit Strg/⌘+N eine neue ActionScript-Datei ❶. Bestätigen Sie den Dialog mit Ok ❷.



◂ **Abbildung 3**
Eine neue ActionScript-Datei anlegen

Anschließend füllen Sie die Datei mit dem noch leeren Klassengerüst.

```
1: package
2: {
3:     import flash.display.MovieClip;
4:     class Spider extends MovieClip
5:     {
6:         public function Spider()
7:         {
8:         }
9:     }
10: }
```

Da die Spinne, der Sie diese Klasse zuweisen werden, ein Movieclip ist und Sie innerhalb der Klasse auf die Movieclip-spezifischen Eigenschaften der Spinne zugreifen wollen, erbt `Spider` in Zeile 4 von `MovieClip`. Diese Klasse muss zu diesem Zweck zuvor (Zeile 3) importiert werden.

Speichern Sie die ActionScript-Datei über Strg/⌘+S unter dem Namen »Spider.as« im Projektordner ab.

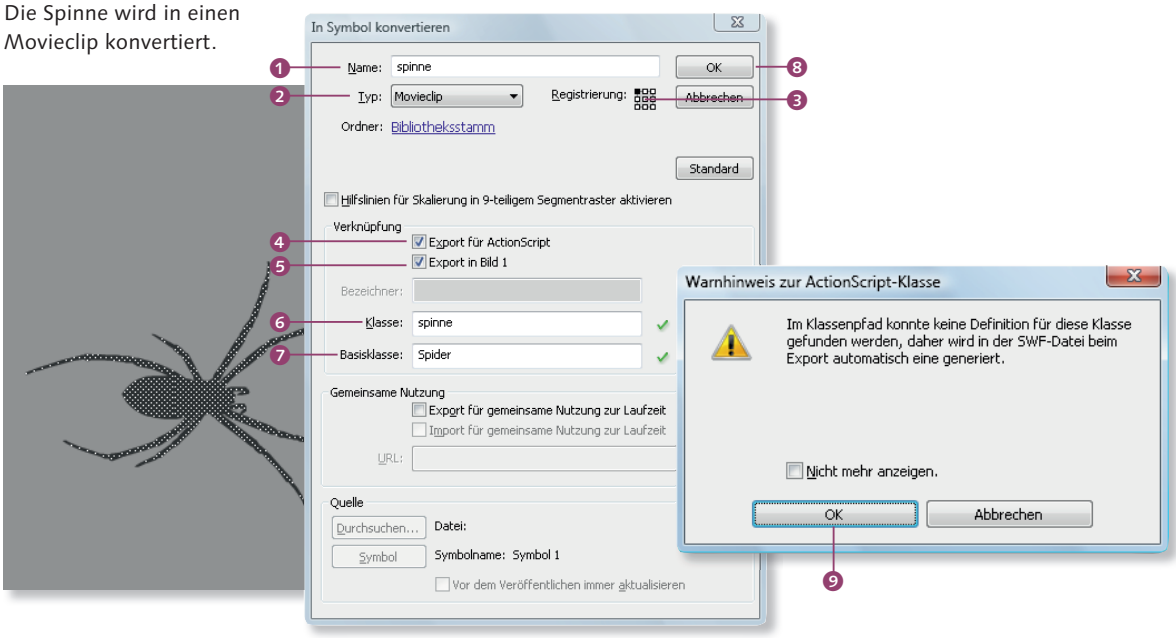
5 Zeichnen der Spinne

Zeichnen Sie eine Spinne mit Hilfe des Stiftwerkzeuges oder machen Sie es sich einfacher und kopieren Sie die Spinnen-Grafik aus der fla-Beispieldatei von der Buch-DVD.

Nach Fertigstellung der Spinnen-Form, wählen Sie diese mit Strg/⌘+A komplett aus und konvertieren diese über F8 in ein Symbol. Im In Symbol Konvertieren-Dialog geben Sie dem Symbol den Namen »spinne« ❶, wählen als Typ Movieclip aus ❷ und ändern die Registrierung auf mittig ❸. Um die Vorbereitungen abzuschließen, müssen Sie noch die Verknüpfung anpassen.

Setzen Sie einen Haken bei Export für ActionScript ❹ – es sollte automatisch ein zweiter Haken bei Export in Bild 1 erscheinen ❺. Danach tragen Sie als Klasse »spinne« ein ❻ und weisen dem Symbol die eben erzeugte Basisklasse »Spider« zu ❼. Bestätigen Sie alle Angaben mit Ok ❽. Den Warnhinweis, dass keine Definition für die Klasse gefunden wurde, bezieht sich auf den von Ihnen zugewiesenen Bezeichner – dieser soll von Flash dynamisch erzeugt werden, so dass Sie die Warnung ohne Bedenken mit Ok ❾ bestätigen können.

Abbildung 4 ▼
Die Spinne wird in einen Movieclip konvertiert.



Anfangsposition der Spinne festlegen 6

Um die Anfangsposition der Spinne bei jedem Aufruf des Flash-Films dynamisch zu verändern, benötigen Sie zwei Variablen, die die maximale x- bzw. y-Position der Spinne speichern. Initialisieren Sie diese direkt oberhalb des Konstruktors in der Spider-Klasse:

```
private var maxX:Number;
private var maxY:Number;
```

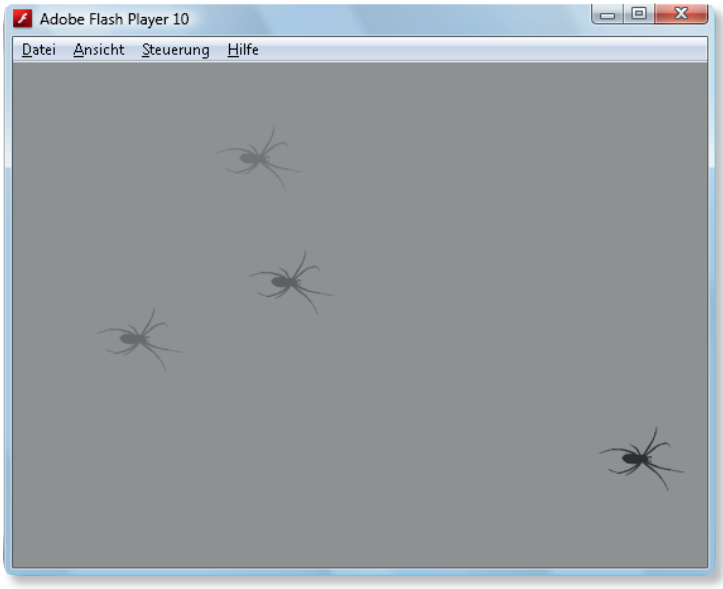
Da die Spinne nicht außerhalb des sichtbaren Bereichs der Bühne platziert werden soll, werden im Konstruktor die beiden Maximalwerte auf die Breite und Höhe der Bühne gesetzt (Zeile 3 und 4).

```
1: public function Spider()
2: {
3:       maxX = stage.stageWidth;
4:       maxY = stage.stageHeight;
5:       x = width/2+Math.random()*(maxX-width);
6:       y = height/2+Math.random()*(maxY-height);
7: }
```

Die beiden darauf folgenden Zeilen berechnen die zufällige x- und y-Position der Spinne mit Hilfe der `Math.random()`-Funktion. Das jeweilige Ergebnis wird der x- und y-Position der Spinne zugewiesen.

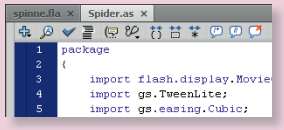
7 Testen des Flash-Films

Speichern Sie die bearbeitete Klassen 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 übereinander gelegt. Diesen Effekt werden Sie in Flash so nicht erzielen.

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



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

Zielposition der Spinne festlegen 8

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

Fügen Sie zu diesem Zweck zwei neue Variablen in die Spider-Klasse ein:

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

Im Anschluss legen Sie eine Methode an, die die neue Spinnen-Position berechnet.

```
private function chooseNewSpot():void
{
   newX = width/2+Math.random()*(maxX-width);
   newY = height/2+Math.random()*(maxY-height);
}
```

Die `chooseNewSpot()`-Methode, lässt die Spinne sich einen neuen Zielpunkt suchen und speichert die Koordinaten in den neu erstellten Variablen `newX` und `newY`. Die Berechnung des Zielpunktes geschieht genau so, wie Sie die Startposition innerhalb des Konstruktors bestimmt haben. Rufen Sie die neue Methode aus dem Klassen-Konstruktor heraus auf:

```
public function Spider()
{
   ...
   chooseNewSpot();
}
```

Bewegen Sie die Spinne 9

Damit sich die Spinne nun vom Startpunkt aus zu dem neu berechneten Zufallspunkt bewegt, benutzen Sie in diesem Schritt die TweenLite-Engine. Die entsprechenden Dateien liegen im GS-Ordner auf der Buch-DVD. Kopieren Sie diese in den im Schritt 1 angelegten Ordner.

Die TweenLite- und die Beschleunigungs-Klasse `Cubic` müssen vor Gebrauch in Ihrer Klasse importiert werden.

```
import gs.TweenLite;
import gs.easing.Cubic;
```

Nach dem Import der benötigten Klassen, können Sie auf alle Funktionen der TweenLite-Tweening-Engine zugreifen. Damit die Spinne sich vom Anfangs- zum Zielpunkt bewegt, »tweenen« – oder einfacher: verändern – Sie die x- und y-Position über einen bestimmten Zeitraum. Hierzu benutzen Sie die `TweenLite.to()`-Funktion. Diese erwartet als ersten Parameter den zu bewegenden Movieclip, als zweiten Parameter die Zeit des Tweens in Sekunden und als letzten Parameter ein Objekt mit den Eigenschaften, die verändert werden sollen, ihren Zielwerten und der Art der Veränderung.

Erstellen Sie eine neue Methode `crawlAway`, in welche Sie anschließend den `TweenLite`-Aufruf durchführen.

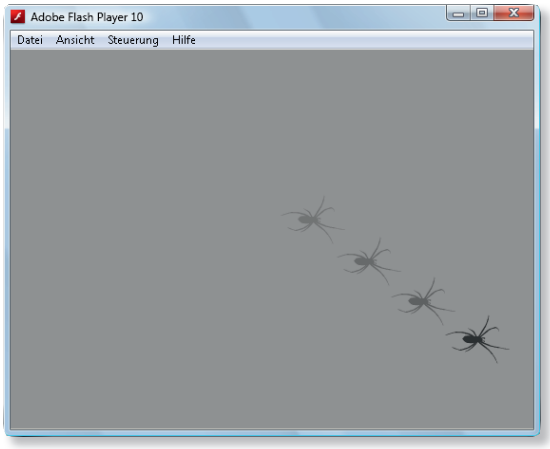
```
private function crawlAway():void
{
   TweenLite.to(this, 3, {x:newX, y:newY, ease:Cubic.easeInOut});
}
```

Noch einmal kurz zur Erklärung: Der erste Parameter der `TweenLite.to`-Funktion `this` bezieht sich auf die Spinne selbst. `3` beschreibt die Dauer des Tweens, also 3 Sekunden. Danach folgt ein Objekt mit den neuen x- und y-Positionen und dem Easing-Typ `Cubic.easeInOut`. »easeInOut« heißt nichts anderes, als dass die Beschleunigung am Anfang leicht zunimmt und gegen Ende der Animation wieder stetig abnimmt. Die `crawlAway()`-Methode rufen Sie aus der `chooseNewSpot()` heraus auf, direkt nachdem die neuen Koordinaten bestimmt worden sind.

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
Diese Klassen sind als Teil der TweenLite-Engine im GS-Ordner enthalten.

10 Testen des Flash-Films

Speichern Sie die Änderungen an der »Spider.as« und testen Sie mit Strg/ ⌘+↵ den neuen Flash-Film.



◂ **Abbildung 6**
Die Spinne bewegt sich von der Startposition zu einer zufälligen zweiten Position. Danach verharrt sie allerdings an dieser Position.

onComplete-Callback hinzufügen 11

Damit die Spinne nicht einfach aufhört zu krabbeln, implementieren Sie in diesem Schritt eine `onComplete`-Callback-Methode, die nach dem Beenden der Animation den Ablauf erneut anstößt.

Hierzu fügen Sie zu der eben erstellten Tween-Anweisung folgenden Parameter ein:

```
TweenLite.to(this, 3, {x:newX, y:newY, ease:Cubic.easeInOut,
onComplete:chooseNewSpot});
```

Bei `onComplete` soll erneut die `chooseNewSpot()`-Funktion aufgerufen werden, so dass die Spinne sich direkt nach Beenden des Tweens ein neues Ziel sucht und sich dort hin bewegt.

Speichern Sie die geänderte Klasse über Strg/⌘+S ab.

Flash-Film erneut testen 12

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 allerdings noch nicht in die Richtung, in die sie krabbelt.

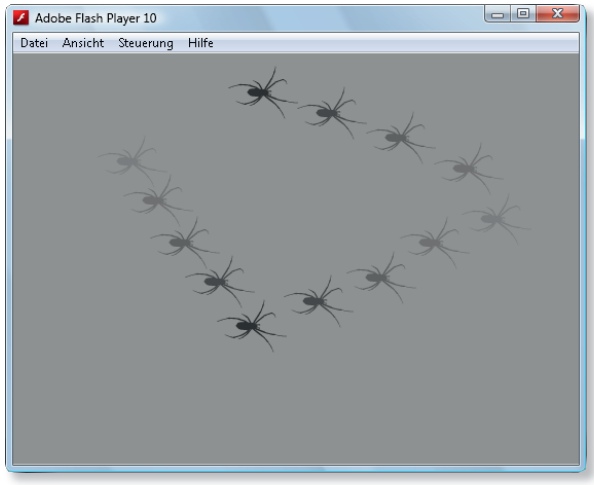


Abbildung 7 ▶
Die Spinne krabbelt fleißig über den Bildschirm.

Rotation der Spinne an die Laufrichtung anpassen 13

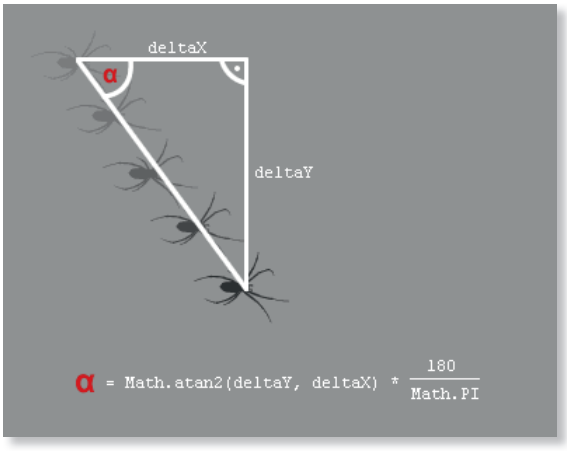
Um die korrekte Drehung der Spinne zu bestimmen, müssen Sie den Winkel ausrechnen, in dem die Spinne sich vom einen Punkt zum nächsten bewegt. Dies ist mit Hilfe der `Math.atan2()`-Funktion innerhalb von Flash sehr leicht.

Die Berechnung des Winkels fügen Sie in die `chooseNewSpot()`-Methode ein, nachdem Sie den neuen Zielpunkt der Spinne bestimmt haben.

Da Sie die Rotation der Spinne nicht einfach so anpassen sollen, werden Sie, genau wie bei der x- und y-Koordinate, einen Tween zur Hilfe nehmen. Hier nun die überarbeitete `chooseNewSpot()` Methode:

```
1: private function chooseNewSpot():void
2: {
3:     newX = width/2+Math.random()*(maxX-width);
4:     newY = 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:     TweenLite.to(this, 1.5, {rotation:degree,
        ease:Cubic.easeInOut, onComplete:crawlAway});
9: }
```

Zeile 3 und 4 sind unverändert. In den zwei darauf folgenden Zeilen wird der x- und y-Abstand des Zielpunktes zum aktuellen Punkt gespeichert. Über die vorher bestimmten Abstände berechnen Sie in Zeile 7 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.



◂ **Abbildung 8**
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 `onComplete`-Callbacks am Ende der Zeile 8 weg.

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

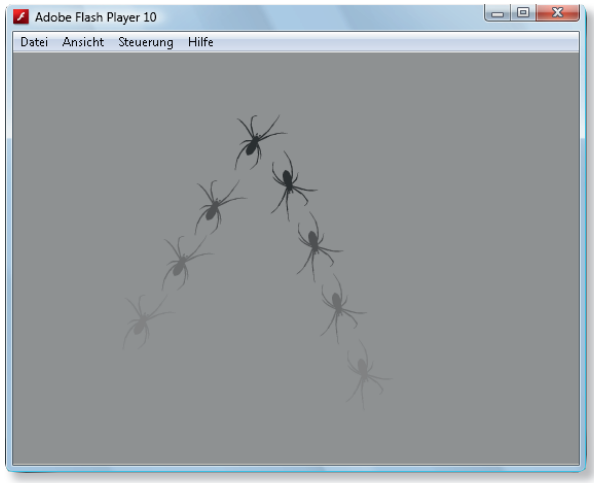
Math.atan2()
Die `Math.atan2()`-Funktion berechnet den Winkel Alpha zwischen zwei Punkten eines rechtwinkligen Dreiecks (siehe Abbildung 8). Sie liefert das Ergebnis im Bogenmaß zurück. Uum diesen Wert in eine Gradzahl umzurechnen, müssen Sie das Resultat mit 180 multiplizieren und anschließend durch Pi dividieren.

Testen des Flash-Films 14

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 dasss sie krabbelt. Dem werden Sie im nächsten Schritt Abhilfe verschaffen.

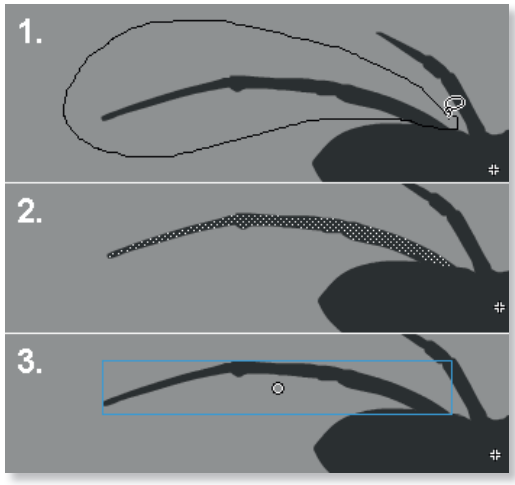
Abbildung 9 ▶
Die Spinne schaut immer in die Richtung, in die sie krabbelt.



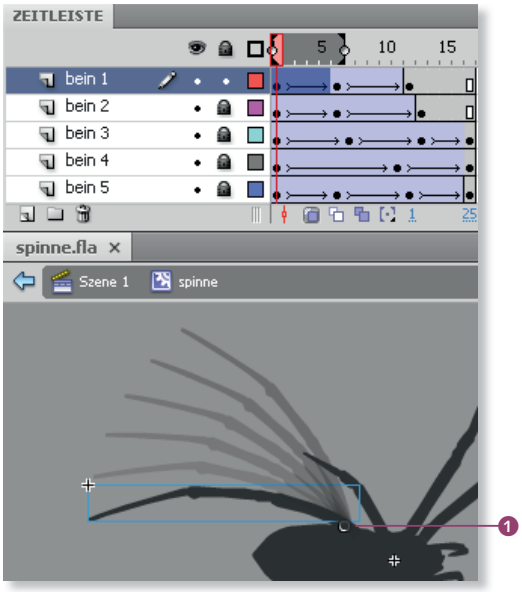
Die Spinnenbeine animieren 15

Wählen Sie innerhalb des Spinnen Movieclips mit Hilfe des Lasso-Werkzeugs 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«.

Abbildung 10 ▶
Jedes Spinnenbein wird in einen Movieclip umgewandelt.



Nachdem Sie die Movieclips erstellt haben, können Sie diese auf der Zeitleiste mit einfachen Bewegungs-Tweens animieren. Hierbei ist es nur wichtig zu beachten, dass Sie die Verbindungsstelle zwischen Körper und Bein als Drehpunkt nehmen. Um dies zu erreichen, müssen Sie den Kreis, der in der vorherigen Abbildung unter 3. noch in der Mitte des Movieclips positioniert ist, mit der Maus auf die in der nächsten Abbildung gekennzeichnet Stelle ❶ ziehen.



◀ **Abbildung 11**
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.

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 Klassen-Variablen – diese speichern die Werte der letzen x- bzw. y-Position der Spinne.

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

Den Konstruktor erweitern Sie dahingehend, dass die Anfangskoordinaten nicht allein gesetzt, sondern auch in den neuen Variablen gespeichert werden (Zeile 5 und 6):

```
1: public function Spider()
2: {
3:     maxX = stage.stageWidth;
4:      maxY = stage.stageHeight;
5:     x = lastX = width/2+Math.random()*(maxX-width);
6:     y = lastY = height/2+Math.random()*(maxY-height);
7:     chooseNewSpot();
8:     }
```

Jetzt fehlt lediglich noch eine Methode, die bei jeder Bewegung der Spinne eine Linie von der letzten zur aktuellen Position der Spinne zeichnet. Hierzu nutzen Sie eine weitere Callback-Funktion der TweenLite-Engine. Den `onUpdate`-Callback fügen Sie dem Bewegungs-Tween in der `crawlAway()`-Methode wie folgt hinzu:

```
TweenLite.to(this, 3, {x:newX, y:newY, ease:Cubic.easeInOut,
onComplete:chooseNewSpot, onUpdate:spinNet});
```

Graphics-Klasse
Jedes Shape-, Sprite- und MovieClip-Objekt enthält eine `graphics`-Eigenschaft, die es ermöglicht, Zeichen-Methoden auszuführen. Wie hier im Beispiel den Faden der Spinne.

Das entstehende Netz zeichnen Sie auf den `root`-Movieclip, so dass es sich nicht mit der Spinne mitbewegt. Die Zeichen-Methoden werden seit Flash CS3 nicht mehr direkt auf den Movieclip angewendet, sondern auf sein jeweiliges Graphics-Objekt, welches Sie als eine Art Leinwand ansehen können.

Importieren Sie also zuerst die benötigte Graphics-Klasse:

```
import flash.display.Graphics;
```

Und hier nun die Methode, die das Netz »spinnt«:

```
1: private function spinNet():void
2: {
3:     var g:Graphics = (root as
           MovieClip).graphics;
4:     g.lineStyle(1, 0xffffff, 0.5);
5:     g.moveTo(lastX, lastY);
6:     g.lineTo(x, y);
7:     lastX = x;
8:     lastY = y;
9: }
```

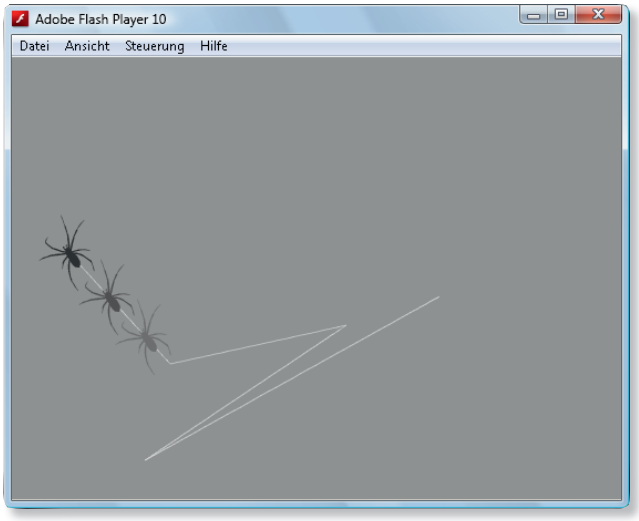
In Zeile 3 speichern Sie die »Leinwand«, auf der Sie zeichnen wollen, in der lokalen Variable `g` zwischen. Der Aufruf in Zeile 4 setzt fest, wie die Linie aussehen soll, die gezeichnet wird. Dabei bestimmt der erste Parameter die Linienstärke in Pixel, der zweite Parameter die Farbe und der dritte die

Transparenz. In der darauf folgenden Zeile wird per `moveTo()`-Funktion der Startpunkt der zu zeichnenden Linie angesprungen. Gezeichnet wird in Zeile 6 – die Linie verläuft dann von `lastX`/`lastY` bis `x`/`y`. In den beiden Zeilen 7 und 8 wird die aktuelle Position der Spinne zwischengespeichert, so dass nicht immer der ganze Faden neu gezeichnet werden muss, sondern nur das Stück, das seit der letzten Bewegung dazugekommen ist.

Speichern Sie die Änderungen mit Strg/⌘+S.

17 Den Flash-Film testen

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.



◂ **Abbildung 12**
Der fertige Flash-Film: ganz schön eklig ;-)