Tobias Hauser, Armin Kappler, Christian Wenz

Das Praxisbuch ActionScript 3

Galileo Press

9 Animation mit ActionScript 3

In Kapitel 5, »Animation«, haben Sie bereits verschiedene Techniken kennengelernt, um Objekte mit Hilfe von Werkzeugen der Entwicklungsumgebung zu animieren. In diesem Kapitel erfahren Sie, wie Sie Animationen mit ActionScript 3 entwickeln.

9.1 Eigenschaften von Anzeigeobjekten

Jedes Anzeigeobjekt, wie ein `Sprite`-, ein `MovieClip`- oder auch ein `Bitmap`-Objekt, besitzt spezifische Instanzeigenschaften, die sich über ActionScript 3 zur Laufzeit ändern lassen. So können Sie z. B. die Position auf der x-Achse eines `Sprite`-Objekts mit dem Instanznamen »mySprite« durch folgenden Aufruf ändern:

Kein Unterstrich in ActionScript 3
In ActionScript 1 und 2 wurde vor jeder Eigenschaft ein Unterstrich geschrieben. Dieser wird in ActionScript 3 nicht mehr verwendet.

```
mySprite.x = 200;
```

In der folgenden Tabelle sind die wichtigsten Instanzeigenschaften aufgelistet.

Eigenschaft	Datentyp	Beschreibung
`alpha`	`Number`	Alphawert von 0 bis 1 (Transparenz)
`x`	`Number`	Position auf der x-Achse
`y`	`Number`	Position auf der y-Achse
`z`	`Number`	Position auf der z-Achse
`height`	`Number`	Höhe des Movieclips
`width`	`Number`	Breite des Movieclips
`scaleX`	`Number`	Skalierung auf der x-Achse
`scaleY`	`Number`	Skalierung auf der y-Achse
`scaleZ`	`Number`	Skalierung auf der z-Achse

◀ **Tabelle 9.1**
Instanzeigenschaften von Anzeigeobjekten

Eigenschaft	Datentyp	Beschreibung
rotation	Number	Rotation (Drehung) in Grad
rotationX	Number	Rotation (Drehung) auf der x-Achse in Grad
rotationY	Number	Rotation (Drehung) auf der y-Achse in Grad
rotationZ	Number	Rotation (Drehung) auf der z-Achse in Grad

▲ **Tabelle 9.1**
Instanzeigenschaften von Anzeigeobjekten (Forts.)

[!] Ein falscher Ansatz: Schleifen
Sollten es Ihnen nicht anders gehen als mir, und Sie denken darüber nach, Schleifen direkt für Animationen einzusetzen, verwerfen Sie den Gedanken möglichst schnell! Schleifen sind zeitunabhängig. Animationen basieren jedoch immer auf Zeit bzw. Zeitleisten. Kein gutes Beispiel:

```
while(mc.x < 100) {
   mc.x +=10;
}
```

Sie werden nur das Ergebnis selbst, nicht aber die Zwischenschritte sehen.

Wenn Sie eine dieser Eigenschaften ändern, geschieht diese Änderung standardmäßig nur einmalig. Um Änderungen an Anzeigeobjekten für Animationen mehrmals bzw. über einen bestimmten Zeitraum durchzuführen, können Sie dafür *Ereignis-Listener*, *Ereignisse* und *Ereignisprozeduren* verwenden.

9.2 Ereignisse

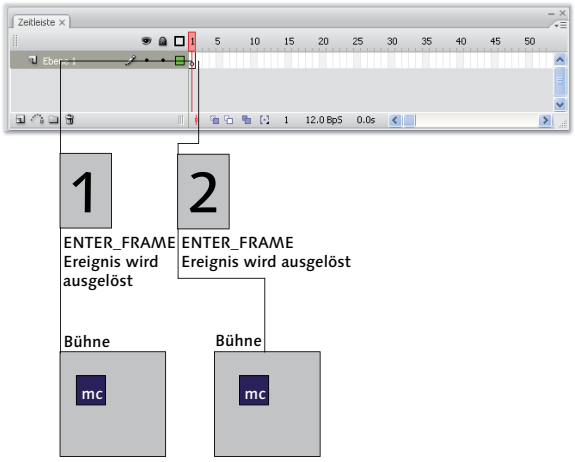
Über bestimmte Ereignisse, Ereignis-Listener und Ereignisprozeduren können Sie Animationen mit ActionScript erstellen. Die wichtigsten werden im Folgenden vorgestellt.

9.2.1 ENTER_FRAME

Das `ENTER_FRAME`-Ereignis tritt wiederholt mit einem Intervall eines Bildes auf. Wie häufig es auftritt, hängt direkt mit der Bildrate des Flash-Films zusammen. Um auf das Auftreten eines Ereignisses zu reagieren, wird ein sogenannter Ereignis-Listener an dem Objekt, auf das sich das Ereignis bezieht, registriert. Wenn Sie beispielsweise ein `Sprite`-Objekt animieren möchten, ist es in den meisten Fällen sinnvoll, einen entsprechenden Ereignis-Listener an diesem `Sprite`-Objekt zu registrieren.

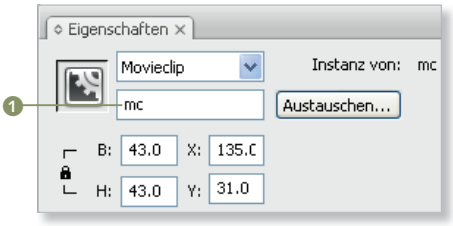
Bei der Registrierung eines Ereignis-Listeners wird eine Ereignisprozedur angegeben, die aufgerufen wird, wenn das Ereignis auftritt. Das `ENTER_FRAME` Ereignis tritt beim Abspielen jedes Bildes auf, also mit einem Intervall von je einem Bild. Eine mögliche Änderung, die Sie mit Hilfe des Ereignisses durchführen, wie z. B. eine Neupositionierung eines Anzeigeobjekts, wird dann anschließend beim Rendern des nächsten Bildes der Zeitleiste sichtbar. Abbildung 9.1 verdeutlicht den zeitlichen Ablauf.

```
mc.addEventListener(Event.ENTER_FRAME,enterFrameHandler);
function enterFrameHandler(e:Event):void {
      e.target.x +=5;
}
```



◀ **Abbildung 9.1**
Zeitlicher Ablauf und Darstellung des Ereignisses ENTER_FRAME

Damit Sie ein Anzeigeobjekt, das über die Entwicklungsumgebung erstellt wurde, über eine Ereignisprozedur steuern können, müssen Sie dem Anzeigeobjekt, im Beispiel einem Movieclip, zuvor einen Instanznamen ❶ im EIGENSCHAFTEN-Fenster zuweisen.



◀ **Abbildung 9.2**
Instanznamen zuweisen

Anschließend können Sie an dem Objekt einen Ereignis-Listener registrieren. Der Listener sorgt dafür, dass auf das Auftreten eines bestimmtes Ereignis »gewartet« und bei Auslösung des Ereignisses die definierte Ereignisprozedur aufgerufen wird.

Die formelle Schreibweise für das Registrieren eines Ereignis-Listeners für ein ENTER_FRAME-Ereignis und eine entsprechende Ereignisprozedur sieht wie folgt aus.

```
mc.addEventListener(Event.ENTER_
FRAME,enterFrameHandler);
function enterFrameHandler(e:Event):void {
   // Anweisung
}
```

Schritt für Schritt: Animation mit Event.ENTER_FRAME-Ereignis

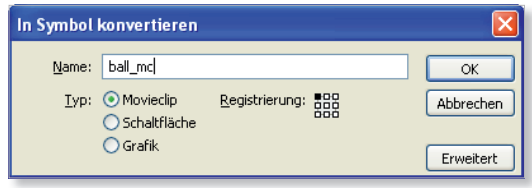
In diesem Workshop erfahren Sie, wie Sie eine Movieclip-Instanz mit Hilfe eines `ENTER_FRAME`-Ereignisses animieren.

1 Flash-Film erstellen

Öffnen Sie einen neuen Flash-Film über das Menü DATEI • NEU. Wählen Sie im Reiter ALLGEMEIN den Dokumenttyp FLASH-DATEI (ACTIONSCRIPT 3.0) aus. Speichern Sie das Dokument über das Menü DATEI • SPEICHERN UNTER in ein beliebiges Verzeichnis unter dem Dateinamen *Animation01.fla* ab.

2 Kreis-Movieclip erstellen

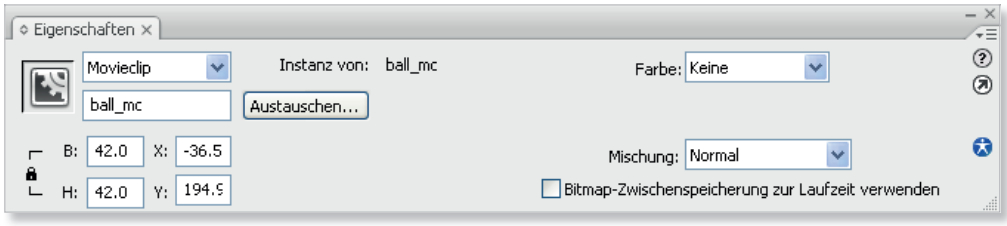
Zeichen Sie mit dem Ellipsen-Werkzeug einen Kreis ein, und wandeln Sie den Kreis über F8 in den Movieclip »ball_mc« um.



▲ **Abbildung 9.3**
In Movieclip konvertieren

3 Instanznamen zuweisen

Wählen Sie den Movieclip auf der Bühne aus, und verschieben Sie ihn nach links auf eine Position außerhalb der Bühne. Weisen Sie ihm im EIGENSCHAFTEN-Fenster den Instanznamen `ball_mc` zu.



▲ **Abbildung 9.4**
Instanznamen zuweisen

4 Aktion zuweisen

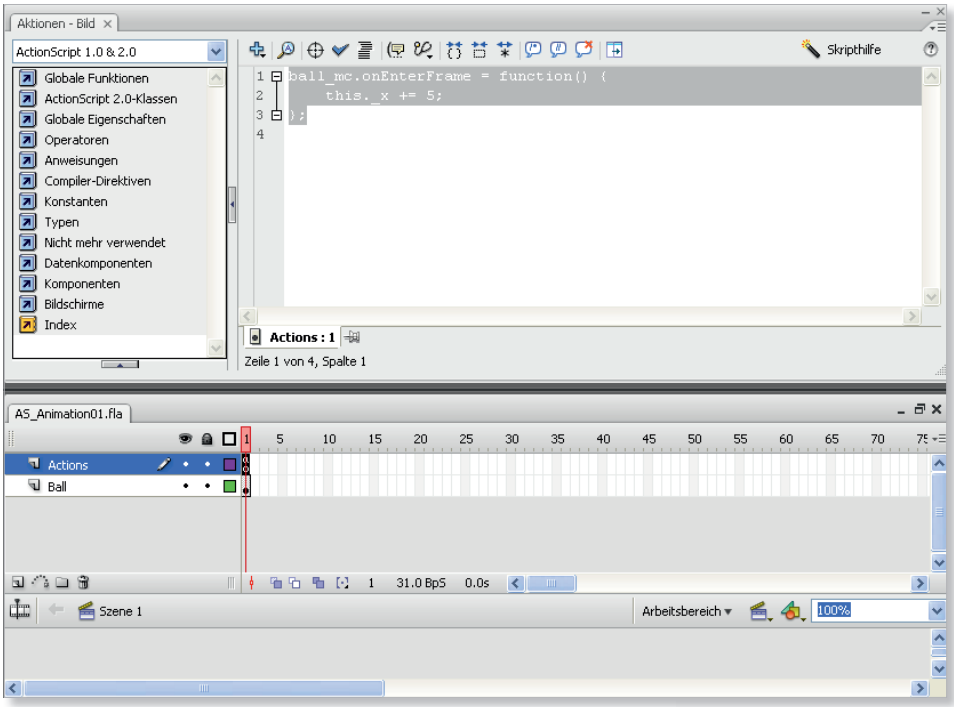
Erstellen Sie eine neue Ebene »Actions«, und weisen Sie dem ersten Schlüsselbild folgenden Code zu:

```
ball_mc.addEventListener(Event.ENTER_FRAME,moveBall);
function moveBall(e:Event):void {
   e.target.x +=5;
}
```

5 Film testen

Testen Sie den Film über Strg/⌘+↵. Der Kreis bewegt sich von links nach rechts. Mit jedem abgespielten Bild wird er um fünf Pixel nach rechts verschoben.

Ergebnis der Übung:
09\ENTER_FRAME\beispiel.fla



▲ **Abbildung 9.5**
Aktion zuweisen

9.2.2 MOUSE_MOVE-Ereignis

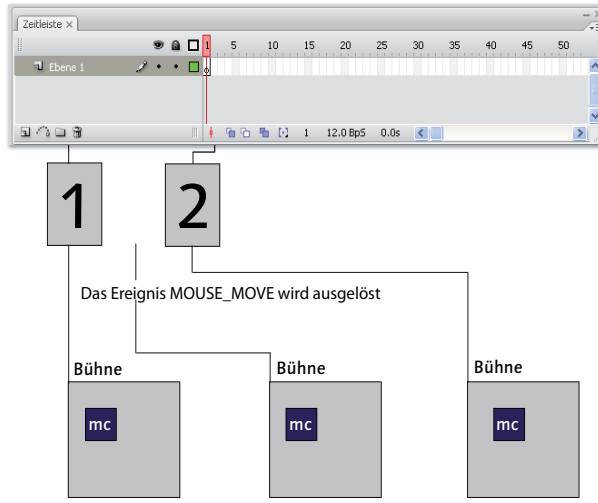
Mit dem Ereignis MOUSE_MOVE steht Ihnen eine weitere Möglichkeit für die Animation eines Objekts per ActionScript zur Verfügung. Das Ereignis wird aufgerufen, wenn die Maus bewegt wird. Dies geschieht unabhängig von der eingestellten Bildrate des Flash-Films. Das Ergebnis, z. B. eine Verschiebung eines Movieclips, ist allerdings erst zu sehen, wenn das nächste Bild des Flash-Films dargestellt wird. Einen Ereignis-Listener für das Ereignis sollten Sie an dem Stage-Objekt registrieren, da er nur aufgerufen wird, wenn sich der Mauszeiger über der Fläche des Objekts, an dem der Ereignis-Listener registriert wird, befindet.

updateAfterEvent

Verwenden Sie die Methode `updateAfterEvent()`; innerhalb der `MOUSE_MOVE`-Ereignisprozedur, um eine sofortige Aktualisierung der Anzeige (Bühne) zu erzwingen. Beachten Sie, dass die Methode eine Methode des übergebenen `MouseEvent`-Objekts ist. Beispiel:

```
function moveBall(e:Mouse
Event):void {
   ...
   e.updateAfterEvent();
}
```

```
mc.addEventListener(MouseEvent.MOUSE_MOVE,mouseMoveHandler);
function mouseMoveHandler(e:MouseEvent):void {
      e.target.x +=5;
```



▲ **Abbildung 9.6**
Zeitlicher Ablauf und Darstellung des Ereignisses `MOUSE_MOVE`

09\Ereignisprozeduren\AS_Animation02.fla

Schritt für Schritt: Animation mit onMouseMove

In diesem Workshop lernen Sie, wie Sie das Ereignis `MOUSE_MOVE` für Animationen einsetzen.

1 Flash-Film öffnen

Öffnen Sie den Flash-Film *AS_Animation02.fla* aus dem Ordner *Ereignisprozeduren*. Ausgangsbasis ist der Movieclip mit dem Instanznamen »ball_mc«. Dieser besitzt einen mittig zentrierten Registrierungspunkt.

2 Aktion zuweisen

Weisen Sie dem ersten Schlüsselbild der Ebene »Actions« folgenden Code zu:

```
stage.addEventListener(MouseEvent.MOUSE_MOVE,moveBall);
function moveBall(e:MouseEvent):void {
   ball_mc.x = mouseX;
   ball_mc.y = mouseY;
   e.updateAfterEvent();
}
```

Immer dann, wenn die Maus bewegt wird, richtet sich der Movieclip anhand der x- und y-Koordinate des Mauszeigers aus.

Mauszeiger ausblenden

Verwenden Sie folgenden Aufruf, wenn Sie den Mauszeiger ausblenden möchten:

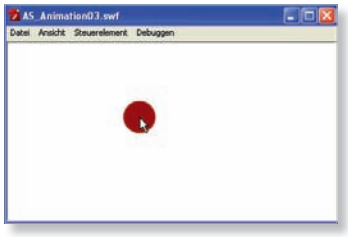
```
Mouse.hide();
```

Zum Einblenden verwenden Sie:

```
Mouse.show();
```

3 Film testen

Testen Sie den Flash-Film über Strg/⌘+↵. Wie Sie vielleicht bemerken, ist der Movieclip zu Beginn nicht zu sehen – erst, wenn Sie die Maus bewegen und die Ereignisprozedur aufgerufen wird, wird er positioniert.



▲ **Abbildung 9.7**
Der Movieclip richtet sich anhand der Maus aus.

4 Movieclip zu Beginn positionieren

Um den Movieclip schon zu Beginn auf den Mauszeiger-Koordinaten zu positionieren, erweitern wir den Code dazu wie folgt:

```
ball_mc.x = mouseX;
ball_mc.y = mouseY;
function moveBall(e:MouseEvent):void {
    ...
}
```

5 Film erneut testen

Testen Sie den Film über Strg/⌘+↵. Der Movieclip wird jetzt zu Beginn einmalig ausgerichtet. ■

Ergebnis der Übung:
09\Ereignisprozeduren\AS_Animation03.fla

9.3 Timer

Die dritte Möglichkeit, eine Instanzeigenschaft mehrmalig für Animationen zu ändern, bietet die sogenannte `Timer`-Klasse. Mit Hilfe der Klasse können Sie eine Funktion definieren, die in einem bestimmten zeitlichen Abstand mehrmalig aufgerufen wird. Auch die Anzahl der Wiederholungen lässt sich festlegen. Die formelle Syntax zur Initialisierung eines `Timer`-Objekts lautet wie folgt:

```
var myTimer:Timer = new Timer(Verzögerung:Number,Wied
erholungen:int);
```

Über den Bezeichner `myTimer` können Sie das Objekt später ansprechen. Über den Parameter `Verzögerung` geben Sie an, in welchem zeitlichen Abstand (in Millisekunden) eine noch zu definierende Funktion wiederholt ausgeführt werden soll. Der Parameter `Wiederholungen` legt fest, wie oft die Funktion ausgeführt werden soll. Der Standardwert ist 0 und führt zu einer unendlichen Wiederholung.

Allein die Initialisierung eines `Timer`-Objekts hat keinerlei Auswirkung. Damit Sie Code über das `Timer`-Objekt ausführen können, müssen Sie einen entsprechenden Ereignis-Listener und eine Ereignisprozedur definieren. Dazu folgendes Beispiel:

Verzögerung und Bildrate

Die tatsächliche Verzögerung hängt auch von der Bildrate des Flash-Films ab. Wenn Sie beispielsweise eine Bildrate von 10 Bildern pro Sekunde eingestellt haben, was pro Bild einen zeitlichen Abstand von 100 Millisekunden entspricht, und Sie die Verzögerung eines `Timer`-Objekts auf 80 Millisekunden setzen, wird die tatsächliche Verzögerung in etwa auch bei 100 Millisekunden liegen.

```
myTimer.addEventListener(TimerEvent.
TIMER,timerHandler);
function timerHandler(e:TimerEvent):void {
   trace("Timer Event triggered");
}
```

Sobald ein `TimerEvent.TIMER`-Ereignis auftritt, wird die Ereignisprozedur `timerHandler` aufgerufen. Um den Timer zu starten, können Sie die Methode `start` des `Timer`-Objekts verwenden:

```
myTimer.start();
```

Über die Methode `stop` können Sie das `Timer`-Objekt jederzeit wieder anhalten:

```
myTimer.stop();
```

Anwendung | Grundsätzlich können Sie ein `Timer`-Objekt ähnlich wie mit einer Ereignisprozedur des Ereignisses `ENTER_FRAME` für Animationen nutzen. Der Hauptunterschied ist, dass die angegebene Ereignisprozedur nicht bildabhängig, sondern tatsächlich zeitabhängig aufgerufen wird. Eine mögliche Änderung einer Instanzeigenschaft, die in der Ereignisprozedur stattfinden könnte, wird allerdings standardmäßig erst angezeigt, wenn das nächste Bild der Zeitleiste gerendert wird (ähnlich wie bei dem Ereignis `MOUSE_MOVE`). Auch hier lässt sich die Methode `updateAfterEvent` nutzen, um eine sofortige Anzeige der Änderung zu erzwingen:

```
var myTimer:Timer = new Timer(100,100);
myTimer.addEventListener(TimerEvent.
TIMER,moveObject);
myTimer.start();
function moveObject(e:TimerEvent):void {
   ball_mc.x +=5;
   e.updateAfterEvent();
}
```

Timer-Objekt löschen

Sollten Sie ein `Timer`-Objekt nicht mehr benötigen, z. B. nachdem Sie den Timer über `stop` angehalten haben, sollten Sie zunächst den Ereignis-Listener entfernen und das Objekt dann auf `null` setzen. Es kann dann beim nächsten Durchlauf des Garbage Collectors aus dem Speicher entfernt werden. Beispiel:

```
...
myTimer.stop();
myTimer.
removeEventListener(Timer-
Event.TIMER,timerHandler);
myTimer = null;
```

9.4 Geschwindigkeit und Beschleunigung

Realistische Animationen
Ein Ansatz, realistische Animationen in Flash über ActionScript zu erstellen, ist die Orientierung an physikalischen Größen der wirklichen Welt.

Die Geschwindigkeit oder, allgemeiner formuliert, die Änderung einer Größe zur Zeit und die Beschleunigung lassen sich in Flash für Animationen einsetzen.

Geschwindigkeit | Die Geschwindigkeit ist eine relativ einfache Größe. Sie lässt sich in ActionScript wie folgt definieren:

```
var vx:uint = 5;
ball_mc.addEventListener(Event.ENTER_FRAME,moveBall);
function moveBall(e:Event):void {
   ball_mc.x +=vx;
}
```

09\Geschwindigkeit_Beschleunigung\AS_Animation01.fla

In diesem Beispiel wird einmalig die Geschwindigkeit definiert und der Variablen `vx` (v = »Velocity«, x = x-Richtung) zugewiesen. Die Eigenschaft `x` wird je Bild mittels einer `ENTER_FRAME`-Ereignisprozedur um die Geschwindigkeit erhöht. Der Movieclip »ball_mc« bewegt sich also von links nach rechts.

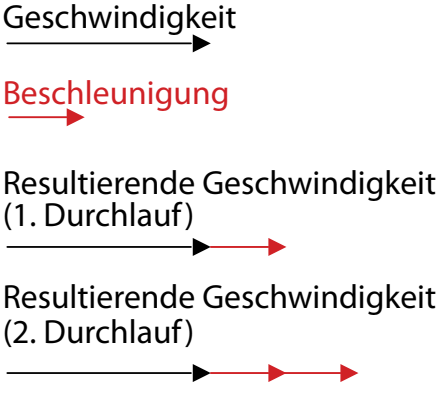
Beschleunigung | Eine beschleunigte Bewegung lässt sich in ActionScript wie folgt definieren:

```
var vx:uint = 5;
var ax:Number = 0.5;
ball_mc.addEventListener(Event.ENTER_FRAME,moveBall);
function moveBall(e:Event):void {
   ball_mc.x +=vx;
   vx+=ax;
}
```

Bewegungsrichtungen
von links nach rechts:
`ball_mc.x+=vx;`
von rechts nach links:
`ball_mc.x-=vx;`
von oben nach unten:
`ball_mc.y +=vy;`
von unten nach oben:
`ball_mc.y -=vy;`

09\Geschwindigkeit_Beschleunigung\AS_Animation02.fla

Die Beschleunigung `ax` (a = »acceleration«, x = x-Richtung) wurde hier zu Beginn auf den Wert `0.5` festgelegt. Die Geschwindigkeit, mit der sich der Movieclip bewegt, wird je Bild durch die Beschleunigung erhöht. Je größer die Beschleunigung ist, desto schneller erhöht sich die Geschwindigkeit dementsprechend.



Kurzschreibweise

Um mehreren Eigenschaften oder Variablen denselben Wert zuzuweisen, können Sie die Kurzschreibweise verwenden:

```
variable0 = variable1 =
Wert;
```

Die entsprechende Langfassung wäre:

```
variable0 = Wert;
variable1 = Wert;
```

◀ **Abbildung 9.8**
Beschleunigung und Geschwindigkeit

09\Geschwindigkeit_Beschleunigung\AS_Animation03.fla

Die Verwendung dieser beiden Größen lässt sich natürlich auch auf jede andere Instanzeigenschaft anwenden. So könnten Sie einen Movieclip über folgenden Code beschleunigt größer werden lassen:

```
var vx:uint = 1;
var ax:Number = 0.5;
ball_mc.addEventListener(Event.ENTER_FRAME,moveBall);
function moveBall(e:Event):void {
   ball_mc.scaleX = ball_mc.scaleY +=vx;
   vx+=ax;
}
```

9.5 Easing

Anwendungsbereich
Animationen, die Easing einsetzen, werden oft als sehr weich, natürlich und rund (»smooth«) empfunden.

Der Begriff *Easing* steht für eine besondere, nicht-lineare beschleunigte oder abgebremste Bewegung. Würde man die Bewegung auf einem Graphen mit zwei Achsen (y-Achse: Zeit, x-Achse: zurückgelegte Strecke) einzeichnen, wäre eine lineare Bewegung eine gerade Linie von links unten nach rechts oben ❶. Eine Kurvenform ❷ entspräche einer nicht-linearen Bewegung.

Es gibt verschiedene Methoden, um Easing für Animationen einzusetzen.

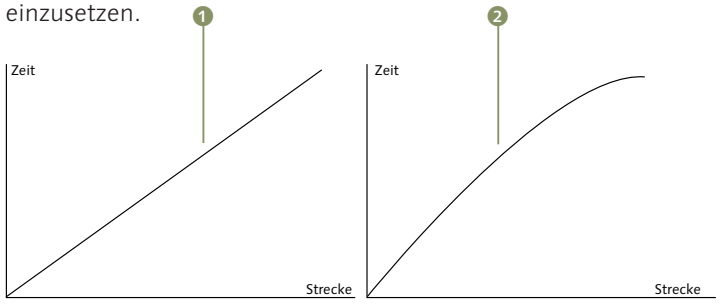


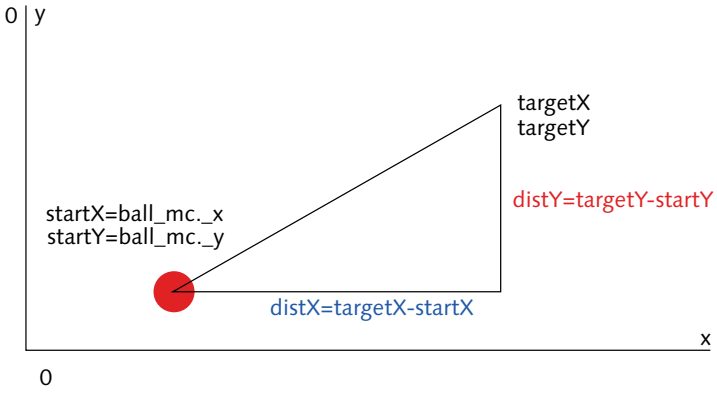
Abbildung 9.9 ▸
Lineare Bewegung ❶, nicht-lineare Bewegung ❷

9.5.1 Bewegung

Eine Bewegung auf zwei Achsen (x, y) lässt sich in Flash in folgende Faktoren auflösen:

- die x- und y-Koordinaten des Startpunkts
- die x- und y-Koordinaten des Ziels

Wenn beispielsweise ein Movieclip das bewegte Objekt ist, entsprechen die x- und y-Koordinaten des Movieclips den Startkoordinaten. Die x- und y-Koordinaten des Ziels könnten z. B. eine fest definierte Position auf der Bühne sein, die Koordinaten eines anderen Anzeigeobjekts oder die Koordinaten des Mauszeigers.



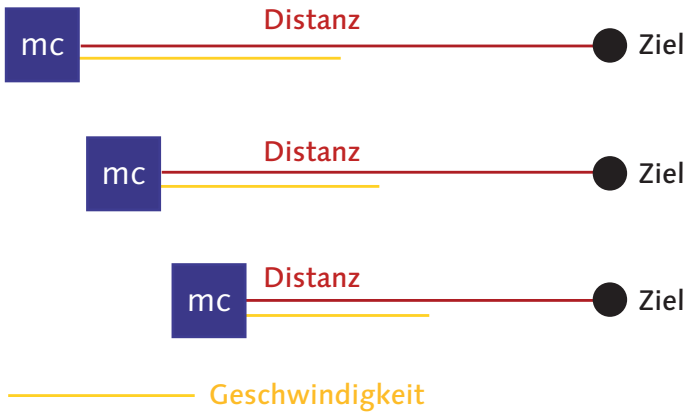
◂ Abbildung 9.10
Berechnung der Distanz zwischen zwei Punkten

Aus den Koordinaten des Startpunkts und den Koordinaten des Zielpunkts lässt sich die Distanz zwischen den beiden Punkten wie folgt ermitteln:

```
var distX:Number = targetX-startX;
var distY:Number = targetY-startY;
```

Wenn sich das Objekt in Zielrichtung bewegt, wird die Distanz immer kleiner. In Flash bedeutet das, dass die Distanz von Bild zu Bild kleiner wird. Dieser Umstand lässt sich nutzen, um ein einfaches Easing für eine Bewegung zu erzielen.

mc._x+=distX/2;



◂ Abbildung 9.11
Easing: Geschwindigkeit in Abhängigkeit von der Distanz zum Zielpunkt

Für ein solches Easing wird zur Änderung der Position des Movieclips ein festgelegter Teil der Distanz z. B. wie folgt verwendet. In diesem Beispiel wird die Distanz durch 4 geteilt, der Movieclip würde sich also je Bild um 1/4 der Distanz verschieben. Sie können hier auch andere Werte, wie z. B. 1/2 oder 1/3, verwenden, die erzielte Wirkung hängt von diesem Wert ab.

Hinweis
Damit die Änderungen nicht nur einmalig durchgeführt werden, können Sie eine der zuvor genannten Methoden verwenden (`ENTER_FRAME`, `MOUSE_MOVE`, Klasse `Timer`).

```
meinMovieClip_mc.x += distX/4;
meinMovieClip_mc.y += distY/4
```

09\Easing_01\AS_Animation01.fla

Schritt für Schritt: Bewegung mit Easing

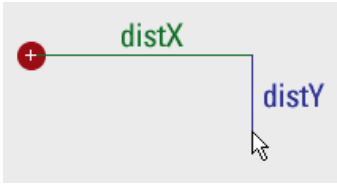
In diesem Workshop lernen Sie, wie Sie eine Movieclip-Instanz mit Hilfe eines Easings animieren.

1 Film öffnen

Öffnen Sie den Flash-Film *AS_Animation01.fla* aus dem Ordner *09\Easing_01*.

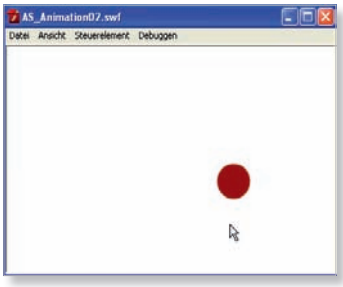
2 Distanz ermitteln

Zunächst wird aus den Startkoordinaten und den Zielkoordinaten, die in diesem Fall den Koordinaten der Maus entsprechen, die Distanz auf der x- und y-Achse berechnet und den Variablen `distX` und `distY` zugewiesen. Fügen Sie dazu auf der Ebene »Actions« folgenden Code ein:



▲ **Abbildung 9.12**
Distanz auf der x- und y-Achse

```
ball_mc.addEventListener(Event.ENTER_FRAME,moveBall);
function moveBall(e:Event):void {
   var startX:Number = ball_mc.x;
   var startY:Number = ball_mc.y;
   var targetX:Number = stage.mouseX;
   var targetY:Number = stage.mouseY;
   var distX:Number = targetX-startX;
   var distY:Number = targetY-startY;
}
```



▲ **Abbildung 9.13**
Der Movieclip folgt dem Mauszeiger.

3 Easing

Sinn und Zweck des Easings ist es, die Bewegung in Abhängigkeit von der Distanz zum Ziel abzubremsen. Dazu ergänzen wir den Code innerhalb der Ereignisprozedur `moveBall` durch folgende Zeilen:

```
ball_mc.x +=distX/4;
ball_mc.y +=distY/4;
```

Der Movieclip wird um ein ¼ der Distanz verschoben; da die Distanz beim nächsten Aufruf der Ereignisprozedur kleiner ist, wird die Verschiebung ebenso immer kleiner.

Ergebnis der Übung:
09\Easing_01\AS_Animation02.fla

4 Film testen

Testen Sie den Film über Strg/⌘+↵. ■

9.5.2 Weitere Instanzeigenschaften animieren

Das zuvor genannte Prinzip des Easings lässt sich in ähnlicher Weise auch für Animationen von anderen Instanzeigenschaften einsetzen. Durch eine allgemeingültigere Formulierung können Sie das Grundprinzip vielseitig verwenden.

Schritt für Schritt: FadeOut mit Easing

In diesem Workshop wird gezeigt, wie Sie eine Movieclip-Instanz mittels eines Easings ausblenden.


09\Easing_02\AS_Animation01.fla

1 Film öffnen

Öffnen Sie den Flash-Film *AS_Animation01.fla* aus dem Ordner *09\Easing_02*.

2 Code zuweisen

Weisen Sie dem ersten Schlüsselbild auf der Ebene »Actions« folgenden Code zu:

```
var easeFaktor:Number = 0.2;
var targetValue:Number = 0;
ball_mc.addEventListener(Event.ENTER_FRAME,
fadeBallOut);
function fadeBallOut(e:Event):void {
   var aktValue:Number = ball_mc.alpha;
   var difValue:Number = targetValue-aktValue;
   ball_mc.alpha += difValue*easeFaktor;
}
```

3 Flash-Film testen

Testen Sie den Film über Strg/⌘+↵. Probieren Sie ruhig einmal verschiedene Werte für den `easeFaktor` aus. Sie bekommen so am besten ein Gefühl dafür, wie sich der Faktor auf die Animation auswirkt. ■


Ergebnis der Übung:
09\Easing_02\AS_Animation02.fla

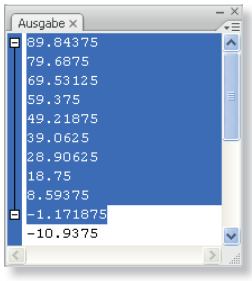
9.5.3 Animation beenden oder loopen

Animationen, die über ActionScript entwickelt werden, sollten entweder geloopt oder, falls das nicht gewünscht ist, beendet werden. Wenn Animationen nicht explizit beendet werden, läuft die Animation, wenn auch vielleicht nicht sichtbar, weiter und benötigt weiterhin unnötigerweise CPU-Leistung und Speicher.

Ereignis-Listener entfernen
Ereignis-Listener, die Sie nicht mehr benötigen, sollten Sie idealerweise über die Methode `removeEventListener` entfernen.

[!] Werte sind oft nicht genau gleich 0
Flash rechnet häufig zwangsweise mit Näherungswerten, so dass Resultate nicht immer ganz exakt sind. Berücksichtigen Sie das, indem Sie statt des Gleichheitsoperators (==) besser kleiner gleich (<=) oder größer gleich (>=) verwenden.
Beispiel:

```
ball_mc.alpha = 1;
ball_
mc.addEventListener(Event.
ENTER_
FRAME,enterFrameHandler);
function
enterFrameHandler(e:Event):
void {
   e.target.alpha -= 0.1;
   trace(e.target.alpha);
}
```



▲ **Abbildung 9.14**
Näherungswerte – oft nicht exakt

*09\Animation_Beenden\AS_Animation01.fl*A

Um eine Animation zu beenden, müssen Sie die Ereignisprozedur entfernen bzw. abmelden. Eine Ereignisprozedur läuft so lange, bis der Movieclip selbst entfernt oder die Ereignisprozedur gelöscht wird. Auch wenn Sie das Ergebnis unter Umständen nicht mehr sehen, wird die Ereignisprozedur weiter aufgerufen und sorgt für eine entsprechende Systemauslastung.

Für den Fall, dass Sie einem Movieclip mit dem Instanznamen »mc« eine `ENTER_FRAME`-Ereignisprozedur zugewiesen haben, würde die Syntax wie folgt lauten:

```
mc.removeEventListener(Event.ENTER_
FRAME,enterFrameHandler);
```

Beachten Sie, dass Sie einen Ereignis-Listener auch innerhalb einer Ereignisprozedur entfernen können:

```
function enterFrameHandler(e:Event):void {
   e.target.removeEventListener(Event.ENTER_FRAME,
   enterFrameHandler);
}
```

Dabei referenziert `e.target` das Objekt, in diesem Fall den Movieclip, an dem der Ereignis-Listener registriert wurde.

Gewöhnlich ist das Beenden einer Ereignisprozedur an eine Bedingung geknüpft. So würde man z. B. eine Movieclip-Instanz entfernen, wenn ihr Alphawert kleiner oder gleich 0 ist. Die Ereignisprozedur sähe dann so aus:

```
function enterFrameHandler(e:Event):void {
   e.target.alpha -= 0.1;
   if(e.target.alpha <= 0) {
      e.target.removeEventListener(Event.ENTER_FRAME,
      enterFrameHandler);
      trace("end");
   }
}
```

Schritt für Schritt: Animation beenden

Dieser Workshop zeigt, wie Sie eine geskriptete Animation beenden.

1 Film öffnen

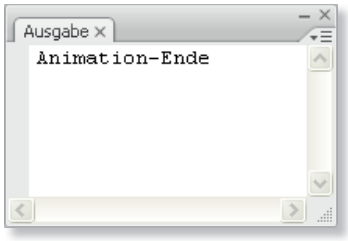
Öffnen Sie den Flash-Film *AS_Animation01.fla* aus dem Ordner *Animation_Beenden*.

2 Code zuweisen

Weisen Sie dem ersten Schlüsselbild der Ebene »Actions« folgenden Code zu:

```
1:   var easeFaktor:Number = 0.2;
2:   var targetXScale:Number = 0;
3:   var targetYScale:Number = 0;
4:   ball_mc.addEventListener(Event.ENTER_
     FRAME,scaleOut);
5:   function scaleOut(e:Event):void {
6:      var aktXScale:Number = e.target.scaleX;
7:      var aktYScale:Number = e.target.scaleY;
8:      var difX:Number = targetXScale-aktXScale;
9:      var difY:Number = targetYScale-aktYScale;
10:      ball_mc.scaleX += difX*easeFaktor;
11:      ball_mc.scaleY += difY*easeFaktor;
12:      if (ball_mc.scaleX<=0.01) {
13:         trace("Animation-Ende");
14:         e.target.removeEventListener(Event.ENTER_
            FRAME,scaleOut);
15:         removeChild(ball_mc);
16:         ball_mc = null;
17:      }
18:   }
```

In Zeile 12 wird geprüft, ob die x-Skalierung des Movieclips kleiner oder gleich 0,01 ist. Wenn das zutrifft, wird die Ereignisprozedur in Zeile 14 gelöscht. Damit Sie sehen können, dass die Bedingung erfüllt ist und die Ereignisprozedur tatsächlich gelöscht wird, wird in Zeile 13 eine Meldung im Ausgabe-Fenster ausgegeben. In Zeile 15 wird das Movieclip-Objekt aus der Anzeigeliste entfernt. In Zeile 16 wird die Referenz auf den Movieclip auf `null` gesetzt, so dass er vom Garbage Collector aus dem Speicher entfernt werden kann.



▲ **Abbildung 9.15**
Die Ereignisprozedur wurde gelöscht.

3 Film testen

Testen Sie den Film über Strg/⌘+↵. ■

Ergebnis der Übung:
09\Animation_Beenden\AS_Animation02.fla

Animation loopen | Das Loopen von geskripteten Animationen ist nicht immer ganz einfach. Auch hier wird mindestens eine Bedingung benötigt, die den Animationsprozess dann jeweils umkehrt. Wenn Sie also z. B. einen Movieclip ausfaden möchten, müssen Sie ihn wieder einfaden, sobald er unsichtbar ist, und umgekehrt. Die formelle Syntax lautet wie folgt:

```
if(mc.alpha >= 1 || mc.alpha <= 0)
   // Kehre den Prozess um
}
```


09\Animation_Loopen\AS_Animation01.fla

Schritt für Schritt: Fading-Animation loopen

In diesem Workshop erfahren Sie, wie Sie eine geskriptete Animation loopen.

1 Film öffnen

Öffnen Sie den Flash-Film *AS_Animation01.fla* aus dem Ordner *Animation_Loopen*.

2 Code zuweisen

Weisen Sie dem ersten Schlüsselbild der Ebene »Actions« folgenden Code zu:

```
var speed:Number = 0.05;
ball_mc.addEventListener(Event.ENTER_FRAME,
enterFrameHandler);
function enterFrameHandler(e:Event):void {
   e.target.alpha -=speed;
   if(e.target.alpha >=1 || e.target.alpha <=0) {
      speed*=-1;
   }
}
```

Sobald der Alphawert des Movieclips größer als 1 oder kleiner als 0 ist, wird die Größenänderung mit –1 multipliziert, wodurch die Animation umgekehrt wird.

3 Film testen


Ergebnis der Übung:
09\Animation_Loopen\AS_Animation02.fla

Testen Sie den Film über Strg/⌘+↵. Die gefadete Animation loopt. ■

9.6 Trigonometrie

Wer eigene Animationsabläufe mit ActionScript entwickeln möchte, wird kaum um einige Grundlagen der Mathematik herumkommen. Viele Ansätze der Mathematik lassen sich sehr gut in die Flash-Welt übertragen, so z. B. auch im Besonderen die Trigonometrie.

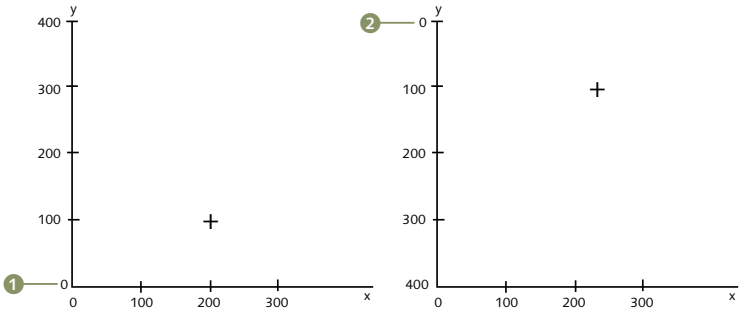
Dieser Teilbereich der Geometrie lässt sich in Flash sehr vielseitig u. a. zur Entwicklung von eindrucksvollen Animationen nutzen. Einige Grundlagen der Trigonometrie und der praktische Einsatz in Flash werden im Folgenden erläutert.

Um die Gesetze der Trigonometrie in Flash praktisch einzusetzen, müssen Sie jedoch zunächst einige Besonderheiten beachten.

Trigonometrie
Bei einer einfachen Anwendung der Trigonometrie, der sogenannten *planen Trigonometrie*, geht es darum, die Beziehung zwischen Seiten eines Dreiecks und dessen Winkeln zu ermitteln.

9.6.1 Koordinatensystem

In der Mathematik wird in der Regel das kartesische Koordinatensystem verwendet, das sich allerdings vom Koordinatensystem von Flash unterscheidet: Bei einem kartesischen Koordinatensystem liegt der Nullpunkt der x-Achse links und der Nullpunkt der y-Achse unten ❶. In Flash verläuft die y-Achse allerdings umgekehrt. So ist der Nullpunkt auf der y-Achse oben ❷. Diese Besonderheit müssen Sie grundsätzlich berücksichtigen, wenn Sie mathematische Regeln in Flash anwenden möchten.

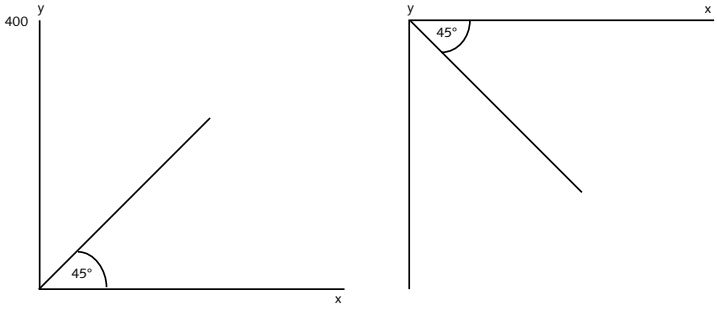


◂ Abbildung 9.16
Koordinatensystem in der Mathematik (links) und in Flash (rechts) mit einem Punkt auf den gleichen Koordinaten

Position eines Movieclips
Ein Movieclip mit den Koordinaten x:0, y:0 befindet sich an der linken oberen Kante der Bühne bzw. seines übergeordneten Anzeigeobjektcontainers.

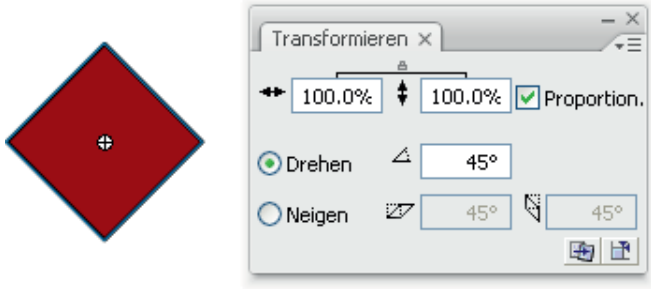
9.6.2 Winkelangabe

Auch die Winkelangabe unterscheidet sich von der in der Mathematik. In der Mathematik wird der Winkel, ausgehend von der x-Achse, gegen den Uhrzeigersinn angegeben – in Flash hingegen im Uhrzeigersinn.



◂ Abbildung 9.17
45-Grad-Winkel in der Mathematik (links) und in Flash (rechts)

Rotationsrichtung | So wird ein Movieclip, sowohl in der Entwicklungsumgebung als auch in ActionScript, der um 45 Grad gedreht wird, nicht nach links, sondern nach rechts gedreht.



▲ **Abbildung 9.18**
Drehung um 45 Grad

9.6.3 Grad- und Bogenmaß – Umrechnung

[Pi]
Die Kreiszahl Pi entspricht in etwa einem Wert von 3,14159. Sie wird häufig in der Geometrie verwendet, um das Verhältnis zwischen dem Durchmesser und dem Umfang eines Kreises zu beschreiben.

Wir sind es gewohnt, Winkel im Gradmaß, also z. B. 90 Grad, anzugeben. Zur Berechnung von Winkeln und Größen wird in ActionScript hingegen das Bogenmaß mit der Maßeinheit »Radiant« verwendet. Keine Sorge – es handelt sich schlichtweg um eine andere Maßeinheit, die leicht umgerechnet werden kann – mehr steckt nicht dahinter.

Ein Winkel von 360 Grad entspricht im Bogenmaß 2 * Pi.

Sie können einen Winkel, der *im Bogenmaß* angegeben ist, wie folgt in ein *Gradmaß* umrechnen:

Grad = Radiant * (180 / Pi)

In ActionScript sieht die Umrechnung des Bogenmaß-Winkels `2*Math.PI` dann wie folgt aus:

```
var radiant:Number = 2*Math.PI;
var grad:Number = radiant*(180/Math.PI);
```

Um einen Winkel, der *in Gradmaß* angegeben ist, *in Bogenmaß umzurechnen*, verwenden Sie folgende Formel:

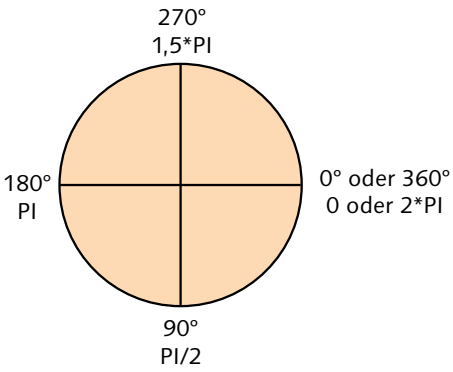
Radiant = Grad * (Pi / 180)

Beispiel:
So wandeln Sie z. B. einen Winkel von 45 Grad in Bogenmaß um:

```
var radiant:Number =
45*(Math.PI/180);
trace(radiant);
```

In ActionScript sieht die Umrechnung des Beispielwinkels 360 Grad dann wie folgt aus:

```
var grad:Number = 360;
var radiant:Number = grad*(Math.PI/180);
```

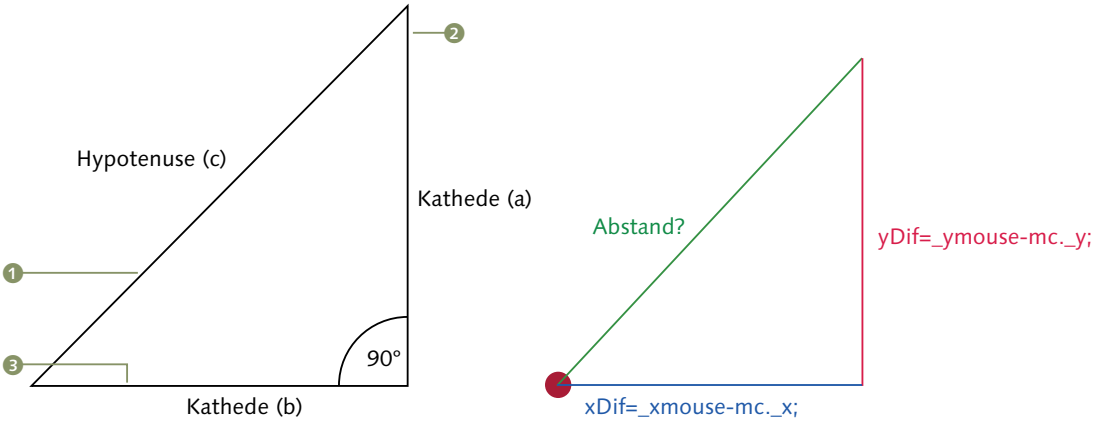


▲ **Abbildung 9.19**
Winkel in Grad- und Bogenmaß

9.6.4 Das rechtwinklige Dreieck

Es ist oft hilfreich, den Abstand zwischen zwei Punkten zu ermitteln. Dazu lässt sich der Satz des Pythagoras einsetzen, nach dem gilt, dass das Quadrat der Hypotenuse ❶ (c), der längsten Seite des Rechtecks, gleich der Summe der Quadrate der jeweils anderen beiden Seiten ❷, ❸ (a, b) ist. Was in textlicher Form kompliziert aussieht, lässt sich mathematisch doch ganz einfach beschreiben. Die gute alte Pythagoras-Formel: $a^2 + b^2 = c^2$

Rechtwinkliges Dreieck
Ein rechtwinkliges Dreieck mit einem 90°-Winkel hat besondere Eigenschaften, die sich für viele verschiedene Animationen über ActionScript nutzen lassen.



▲ **Abbildung 9.20**
Berechnung des Abstands zwischen zwei Punkten

Erinnern Sie sich noch? Die Formeln zum Berechnen der Länge der beiden Katheten haben Sie bereits im Abschnitt über Easing kennengelernt.

Wenn der Abstand zwischen einem Movieclip mit dem Instanznamen »mc« und der Mausposition zu ermitteln wäre, wäre demnach die Länge der Kathete a wie folgt zu errechnen:

```
var yDif:Number = stage.mouseY - mc.y;
```

Reihenfolge der Berechnung
Wie auch ein Taschenrechner der mathematischen Regel folgt, werden zuerst Multiplikationen (*) und Divisionen (/) und dann gegebenenfalls die Addition (+) und Subtraktion (–) durchgeführt. Sie brauchen hier also nicht extra Klammern zu setzen.

Negative Koordinaten

Beachten Sie, dass Koordinaten und Abstände in Flash grundsätzlich auch negative Werte sein können. Bei der Berechnung der Distanz ist das unerheblich, da die Distanz, z. B. `xDif`, mit sich selbst multipliziert wird. Das Ergebnis ist also immer positiv, da –2 * –2 = 4 ist.

Und die Länge der Kathete b:

```
var xDif:Number = stage.mouseX - mc.x;
```

Stellt man den Satz des Pythagoras etwas um, lässt sich die Distanz zwischen zwei Punkten wie folgt berechnen:

```
var distanz:Number = Math.sqrt(xDif*xDif+yDif*yDif);
```

Dabei entspricht der Ausdruck `Math.sqrt` »der Wurzel aus« (`sqrt` = engl. »square root«).

Seiten und Winkel eines rechtwinkligen Dreiecks | Bei einem Dreieck ist die Summe der Winkel immer gleich 180°. Bei einem rechtwinkligen Dreieck ist der größte Winkel der 90°-Winkel, dem gegenüber die Hypotenuse liegt. Wenn Sie sich auf einen anderen der beiden Winkel beziehen, spricht man von der dem Winkel gegenüberliegenden Seite als *Gegenkathete* und von der am Winkel anliegenden Seite als *Ankathete*.

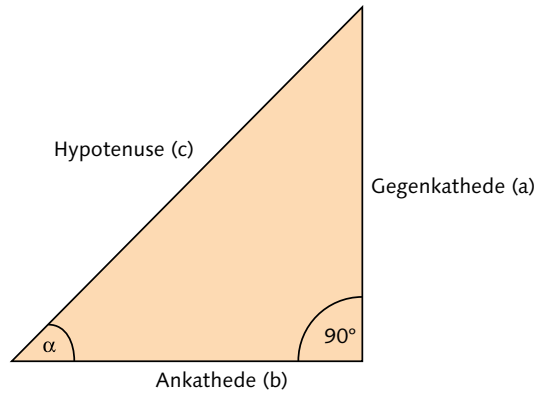


Abbildung 9.21 ▸
Seiten eines Dreiecks in Bezug auf den Winkel α

Die Seiten und Winkel eines rechtwinkligen Dreiecks stehen in einem besonderen Verhältnis zueinander, das sich in Flash, wie Sie später noch sehen werden, für Animationen nutzen lässt.

9.6.5 Schwingende Bewegung

Richtung der Schwingung

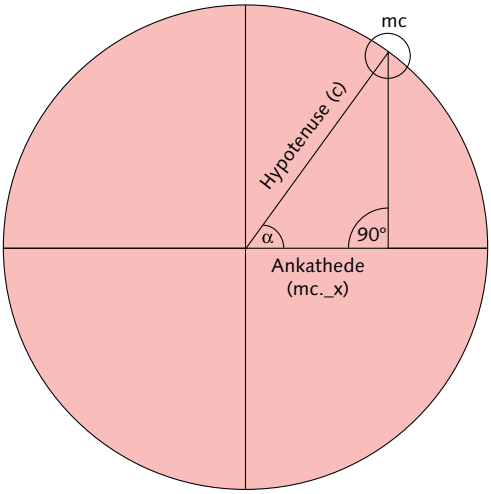
Eine Schwingung in Richtung der x-Achse kann über die Kosinus-Funktion erreicht werden und eine Schwingung in Richtung der y-Achse über die Sinus-Funktion.

Mit Hilfe der trigonomischen Kosinus- und Sinus-Funktion können Sie ein Objekt auf unterschiedliche Weise animieren. So können Sie ein Objekt über die Funktionen in eine schwingende Bewegung versetzen.

Kosinus | In einem rechtwinkligen Dreieck gilt die Formel:
cos α = *Ankathete / Hypotenuse*

Die Ankathete entspricht der Distanz auf der x-Achse eines Punktes zum Mittelpunkt eines Kreises. Für die x-Position muss also die Ankathete berechnet werden. Stellt man die Formel danach um, so erhält man:

*Ankathete = cos α * Hypotenuse*



▲ **Abbildung 9.22**
Berechnung der x-Koordinate des Movieclips mit dem Instanznamen »mc«

Der Winkel verläuft in einem Kreis von 0 bis 360 Grad. Die Hypotenuse ist der Radius des Kreises. Bei einem Movieclip mit dem Instanznamen »mc«, der auf einem Kreis positioniert wird, könnte folgende Formel zur Berechnung der Position auf der x-Achse, in Abhängigkeit vom angegebenen Winkel, angewandt werden.

```
mc.x = Math.cos(winkel)*radius;
```

Schritt für Schritt: Schwingende Bewegung auf der x-Achse

AS_Animation\SchwingungX\SchwingungX_01.fla

In diesem Workshop lernen Sie, wie Sie eine schwingende Bewegung mit Hilfe der Kosinus-Funktion auf der x-Achse erzielen.

1 Film öffnen

Öffnen Sie den Flash-Film *SchwingungX_01.fla* aus dem Ordner *09\SchwingungX*. Ausgangsbasis ist ein Kreis, der in einen Movieclip mit dem Instanznamen »kreis_mc« umgewandelt wurde. Der Registrierungspunkt der Kreises ist mittig. Der Movieclip wurde mittig auf der Bühne platziert.

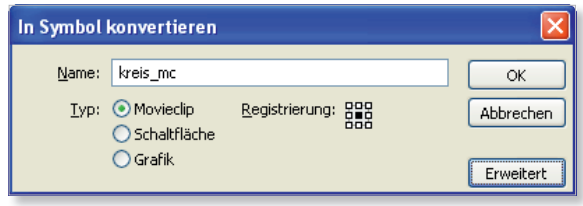


Abbildung 9.23 ▸
Mittig registrierter Movieclip

2 Code zuweisen

Weisen Sie dem ersten Schlüsselbild auf der Ebene »Actions« folgenden Code zu:

```
var winkel:Number = 0;
var radius:Number = 200;
kreis_mc.addEventListener(Event.ENTER_FRAME,
moveBall);
function moveBall(e:Event):void {
   kreis_mc.x = Math.cos(winkel)*radius;
   winkel += 0.1;
};
```

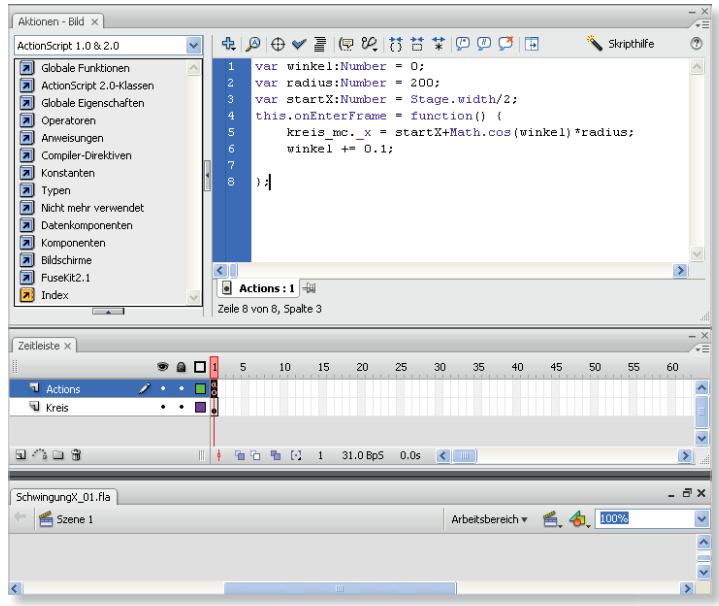


Abbildung 9.24 ▸
Aktionen zuweisen

Zwischenergebnis der Übung: *09\SchwingungX\SchwingungX_02.fla*

3 Film testen

Testen Sie den Film über Strg/⌘+↵. Der Movieclip schwingt auf der x-Achse mit einem Radius von 200 Pixeln hin und her. Dabei ist der Mittelpunkt der Schwingung der linke Rand der Bühne (x:0), und das unabhängig davon, wo der Movieclip zunächst positioniert wurde.

4 Mittelpunkt definieren

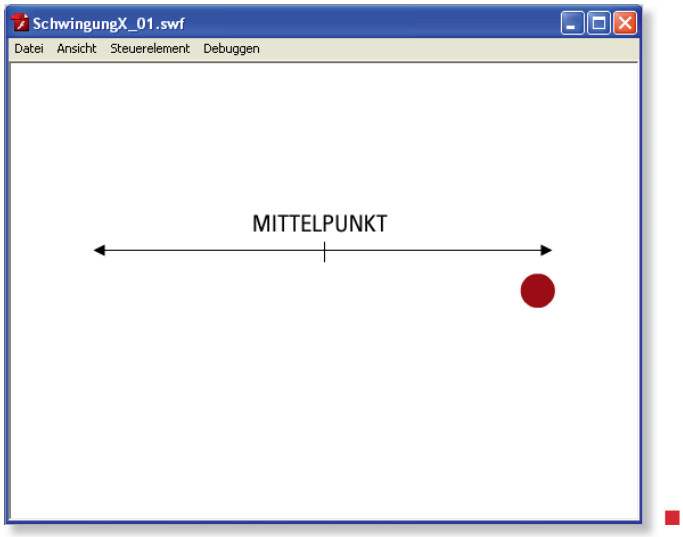
Ändern Sie den Code wie folgt:

```
…
var startX:Number = stage.stageWidth/2;
function moveBall(e:Event):void {
   kreis_mc.x = startX+Math.cos(winkel)*radius;
   winkel += 0.1;
};
```

Der Mittelpunkt wurde in diesem Fall jetzt auf die Hälfte der Bühnebreite, d. h. auf die Mitte der Bühne, festgelegt.

5 Film erneut testen

Testen Sie den Film über Strg/⌘+↵. Ändern Sie ruhig einmal die Werte für den Mittelpunkt (z. B. `100`) und die Steigerung des Winkels (z. B. `0.5`), und beobachten Sie die Auswirkungen.



Ergebnis der Übung:
09\SchwingungX\SchwingungX_03.fla

◂ **Abbildung 9.25**
Der Bewegungsbereich des Movieclips

Sinus | In einem rechtwinkligen Dreieck gilt die Formel:
sin α = Gegenkathete/Hypotenuse

Die Gegenkathete entspricht der Distanz auf der y-Achse eines Punktes zum Mittelpunkt eines Kreises. Für die y-Position muss also die Gegenkathete berechnet werden. Stellen wir die Formel dementsprechend um, erhalten wir:
*Gegenkathete = sin α *Hypotenuse*

Positionierung auf der y-Achse
Bei einem Movieclip »mc«, der auf einem Kreis positioniert wird, lässt sich demnach folgende Formel zur Positionierung auf der y-Achse anwenden:

```
mc.y = Math.
sin(winkel)*radius;
```

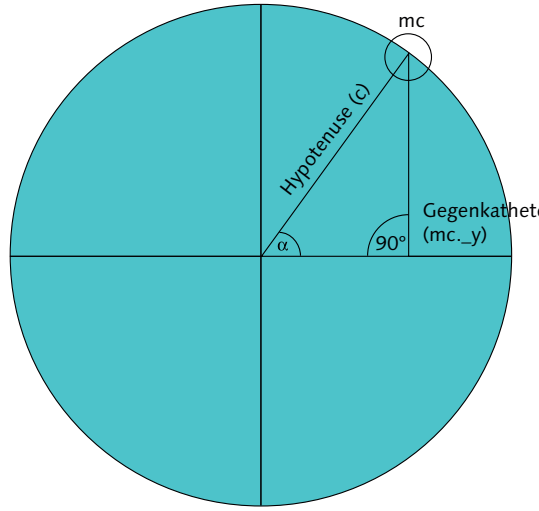


Abbildung 9.26 ▶
Berechnung der y-Koordinate des Movieclips mit dem Instanznamen »mc«

Schritt für Schritt: Schwingende Bewegung auf der y-Achse

In diesem Workshop wird gezeigt, wie Sie eine schwingende Bewegung auf der y-Achse erzielen.

09\SchwingungY\SchwingungY_01.fla

1 Film öffnen

Öffnen Sie den Flash-Film *SchwingungY_01.fla* aus dem Ordner *SchwingungY*. Ausgangsbasis ist ein Kreis, der in einen Movieclip mit dem Instanznamen »kreis_mc« umgewandelt wurde. Der Registrierungspunkt des Kreises ist mittig.

2 Code zuweisen

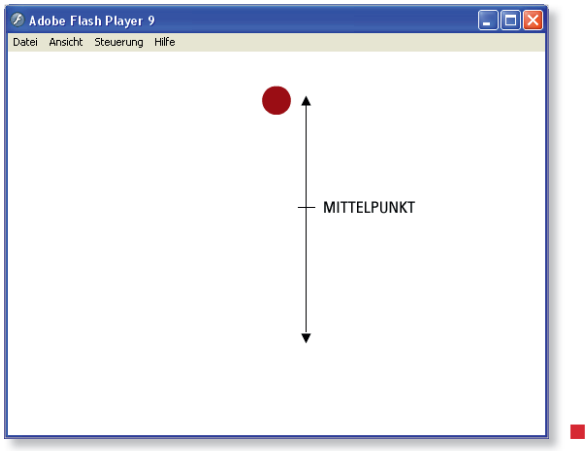
Weisen Sie dem ersten Schlüsselbild auf der Ebene »Actions« folgenden Code zu:

```
var winkel:Number = 0;
var radius:Number = 150;
var startY:Number = stage.stageHeight/2;
kreis_mc.addEventListener(Event.ENTER_FRAME,moveBall);
function moveBall(e:Event):void {
   kreis_mc.y = startY+Math.sin(winkel)*radius;
   winkel += 0.1;
};
```

Ergebnis der Übung:
09\SchwingungY\SchwingungY_02.fla

3 Film testen

Testen Sie den Flash-Film über Strg/⌘+↵. Der Movieclip schwingt auf der y-Achse mit einem Radius von 150 Pixeln um den Mittelpunkt (`startY`).



◀ **Abbildung 9.27**
Der Bewegungsbereich des Movieclips

3D Bewegung mit Hilfe der z-Achse | Mit ActionScript 3 bzw. Flash 10 wurde die z-Achse eingeführt. Früher musste man diese Achse durch Skalierung selbst simulieren, um 3D-Animationen annähernd realistisch abzubilden. Das ist jetzt nicht mehr notwendig. Sie müssen allerdings beachten, dass die z-Achse tatsächlich keine echte dritte Dimensionen ist, wie man es aus 3D-Programmen kennt, sondern eher als »2,5te Dimension« bezeichnet werden kann, da keine automatische Tiefenverwaltung integriert ist. Das bedeutet: Objekte die sich z. B. entgegengesetzt auf der z-Achse bewegen, verändern nicht ihre Tiefe, nachdem sie aneinander vorbeigezogen sind.

```
mc2.addEventListener(Event.ENTER_FRAME,moveObject);
function moveObject(e:Event):void {
      e.target.z+=5;
}
```

▲ **Abbildung 9.28**
Der Movieclip »mc2« bewegt sich auf der z-Achse nach hinten. Die Tiefe des Movieclips »mc2« ändert sich nicht automatisch, wenn er den Movieclip »mc1« passiert.

Eine Änderung der Tiefe, d. h. die Änderung des `childIndex` eines Anzeigeobjekts in der Anzeigeliste, müssen Sie immer noch selbst durchführen. Der folgende Workshop zeigt eine praktische Anwendung der z-Achse und demonstriert, wie Sie die Tiefe eines Objekts entsprechend ändern können, so dass der Eindruck einer 3D-Bewegung entsteht.

[!] Registrierungspunkt beachten
Auch bei der z-Achse ist der Registrierungspunkt zu beachten. Wenn Sie beispielsweise einen Kreis oder ein Rechteck auf der z-Achse bewegen möchten, sollten Sie den Registrierungspunkt des Movieclips auf mittig stellen. Ist der Registrierungspunkt eines Movieclips links oben, bewirkt eine Änderung der z-Eigenschaft eine Skalierung und eine Verschiebung auf der x- und y-Achse.

09\3D_Bewegung\3D_Bewegung_01.fla

Schritt für Schritt: 3D-Bewegung und Tiefenänderung

In diesem Workshop wird erläutert, wie Sie die z-Achse für eine 3D.Bewegung einsetzen und die Tiefe eines Movieclips passend zur Situation ändern.

1 Flash-Film öffnen

Öffnen Sie den Flash-Film *3DBewegung_01.fla* aus dem Verzeichnis *3D_Bewegung*. In dem Flash-Film wurden zwei Movieclips »erde_mc« und »mond_mc« und eine Hintergrundgrafik angelegt.

2 Variablen und Bewegung iniitieren

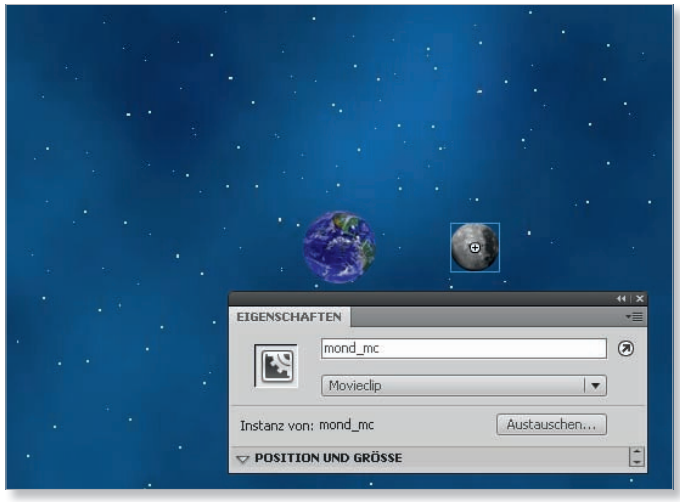


Abbildung 9.29 ▸
Die Ausgangssituation

Weisen Sie dem ersten Schlüsselbild der Ebene »Actions« zunächst folgenden Code zu:

```
1:   mond_mc.addEventListener(Event.ENTER_FRAME,
     moveObject);
2:   var winkel:Number = 0;
3:   var radius:uint = 200;
4:   var startX:Number = stage.stageWidth/2;
5:   var startY:Number = stage.stageHeight/2;
6:   function moveObject(e:Event):void {
7:      e.target.x = startX + Math.cos(winkel)
       *radius;
8:      e.target.y = startY+Math.sin(winkel)*20;
9:      e.target.z = Math.sin(winkel)*radius;
10:      winkel += 0.1;
11:  }
```

In Zeile 1 wird ein Ereignis-Listener registriert, der mit einem Intervall von einem Bild die Funktion `moveObject` aufruft. Anschließend wird in Zeile 2 die Variable `winkel` definiert, und ihr wird der Startwert 0 zugewiesen. Über die Variablen `startX` und `startY` definieren Sie den Mittelpunkt der kreisförmigen 3D-Bewegung (Zeile 4, 5). Innerhalb der Funktion `moveObject` wird der Movieclip »mond_mc« zunächst auf der x-Achse bewegt, siehe dazu auch den Abschnitt »Schwingung in x-Richtung«. Der Code in Zeile 8 sorgt für die Bewegung auf der y-Achse, siehe dazu auch den Abschnitt »Schwingung in y-Richtung«.

Der Mond kreist tatsächlich nicht ganz parallel zur Erdachse. Um eine entsprechende Bewegung auf der y-Achse zu simulieren, wird der Wert hier mit 20 multipliziert (Zeile 8). Der Faktor gibt die Auslenkung der Bewegung auf der y-Achse an. Um eine 3D-Bewegung zu simulieren, wird der Wert der Eigenschaft `z` des Movieclips größer und wieder kleiner, das heißt, der Movieclip wird größer und kleiner. Das geschieht mit einer Sinus-Funktion (Zeile 9). Der Wert der Variablen `winkel` wird je Durchlauf um `0.1` erhöht. Wenn Sie den Wert erhöhen, wird die Bewegung schneller; verringern Sie den Wert, erhalten Sie eine langsamere Bewegung.

3 Flash-Film testen

Testen Sie den Flash-Film über Strg/⌘+↵. Wie Sie sehen, wirkt die Animation noch nicht realistisch, da sich die Tiefe des Movieclips »mond_mc« bisher nicht verändert. Der Movieclip »mond_mc« bleibt visuell immer auf einer Ebene vor dem Movieclip »erde_mc«, auch dann, wenn er sich nach der z-Achse hinter dem Movieclip »erde_mc« befindet.

◂ Abbildung 9.30
Noch wirkt die Animation nicht realisitisch.

4 Tiefe des Movieclips je nach Position ändern

Ändern Sie den Code innerhalb der Funktion `moveObject` wie folgt (Änderungen sind fettgedruckt):

```
1:   function moveObject(e:Event):void {
2:      e.target.x = startX + Math.cos(winkel)
        *radius;
3:      e.target.y = startY+Math.sin(winkel)*20;
4:      e.target.z = Math.sin(winkel)*radius;
5:      winkel += 0.2;
6:      if(winkel > Math.PI/4) {
7:         setChildIndex(DisplayObject(e.target),1);
8:      }
9:      if(winkel > Math.PI) {
10:         setChildIndex(DisplayObject(e.target),2);
11:      }
12:      if(winkel > Math.PI*2) {
13:         winkel = 0;
14:      }
15:   }
```

Wenn der Winkel größer ist als `Math.PI/4`, wird der Movieclip »mond_mc« auf dem `ChildIndex` 1 platziert (Zeile 6–8). `Math.PI/4` entspricht dem Achtel von 360 Grad – also 1/8 einer Umdrehung. Dann befindet er sich auf dem Weg hinter die Erde (gestrichelte Linie). Er wird also auf die Ebene hinter die Erde platziert. Sobald der Winkel größer als `Math.PI` ist, wird der Movieclip auf den ChildIndex 2 platziert, eine Ebene vor der Erde (durchgezogene Linie). Nach einer Umrundung (entspricht `Math.PI*2`) wird der Wert der Variablen `winkel` wieder auf 0 gesetzt (Zeile 12–14).

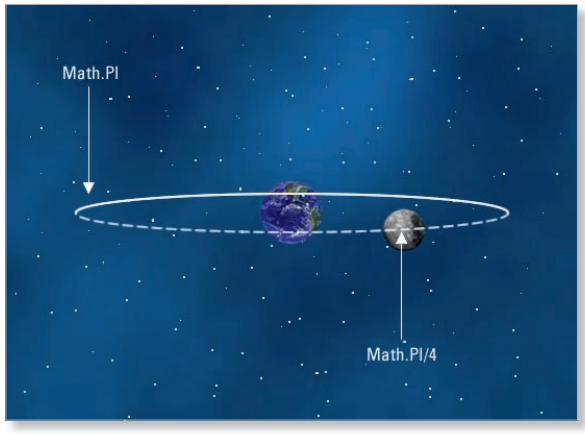


Abbildung 9.31 ▸
Die Tiefe wird situationsabhängig geändert.

5 **Fertig! Flash-Film testen**

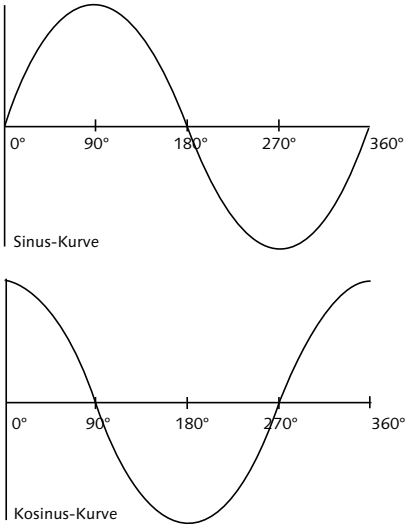
Testen Sie den Flash-Film über Strg/⌘+↵. Die Animation wirkt dank der Änderung der Tiefe jetzt schon deutlich realistischer. ■

Ergebnis der Übung:
09\3D_Bewegung\3D_Bewegung_02.fla

9.6.6 Kreisbewegung

Im Prinzip haben Sie jetzt bereits alles Notwendige kennengelernt, um eine Kreisbewegung über ActionScript zu animieren. Zum besseren Verständnis der Zusammenhänge folgt noch eine zusätzliche visuelle Erläuterung.

Sie kennen wahrscheinlich den Graphen einer Sinus- und einer Kosinus-Kurve. Die Form der Graphen zeigt jeweils das Resultat der jeweiligen Funktion mit Winkeln von 0 bis 360 Grad.

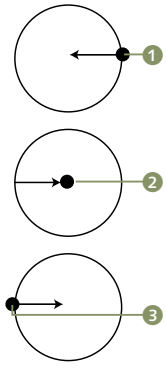


◂ **Abbildung 9.32**
Sinus- und Kosinus-Kurve

Visualisierung der Kreisbewegung in x- und y-Richtung | Versuchen Sie einmal, die Bewegung eines Objekts auf einem Kreis in x- und y-Richtung separat zu betrachten. Probieren Sie es zunächst mit der x-Richtung. Stellen Sie sich dabei z. B. einen Kreis mit einem Radius von 100 Pixeln vor. Angenommen, die Position des Objekts ist zu Beginn rechts in der Mitte des Kreises ❶. Auf der x-Achse befindet sich das Objekt zunächst also auf x = 100. Dann bewegt sich das Objekt nach links zum Mittelpunkt des Kreises. x wird also kleiner, bis es 0 ist ❷. Vom Mittelpunkt der x-Achse geht es weiter in den negativen Bereich ❸ usw.

Dieselbe Visualisierung lässt sich ebenso für die y-Bewegung mittels der Sinus-Kurve anwenden. Sie können eine Kreisbewegung also mit Sinus und Kosinus in ihre x- und y-Achse separieren.

Tipp
Vergleichen Sie diese Bewegung mit dem Verlauf der Kosinus-Kurve – sie ist »überraschenderweise« identisch.



▲ **Abbildung 9.33**
Bewegung auf der x-Achse

Anwendung für eine Kreisbewegung | Die Auflösung der Bewegung in x- und y-Richtung lässt sich in Flash nutzen, um eine Kreisbewegung zu erzielen. Dazu rufen Sie die Ihnen bereits bekannten Kosinus- und Sinus-Funktionen einfach gleichzeitig auf:

```
mc.x = Math.cos(winkel)*radius;
mc.y = Math.sin(winkel)*radius;
```


09\Kreis_Elliptische_Bewegung\Kreisbewegung01.fla

Schritt für Schritt: kreis- und ellipsenförmige Bewegung

In diesem Workshop lernen Sie, wie Sie eine kreis- und eine ellipsenförmige Bewegung eines Objekts erzielen.

1 Flash-Film öffnen

Öffnen Sie den Flash-Film *Kreisbewegung01.fla* aus dem Ordner *Kreis_Elliptische_Bewegung*.

2 Anfangswerte initialisieren

Weisen Sie dem ersten Schlüsselbild auf der Ebene »Actions« folgenden Code zu:

```
var winkel:Number = 0;
var radiusX:Number = 150;
var radiusY:Number = 150;
var startX:Number = stage.stageWidth/2;
var startY:Number = stage.stageHeight/2;
```

3 Ellipsenförmige Bewegung

Um eine elliptische Bewegung zu erreichen, müssen Sie nur die Radien ändern, z. B.:

```
var radiusX:Number = 150;
var radiusY:Number = 20;
```

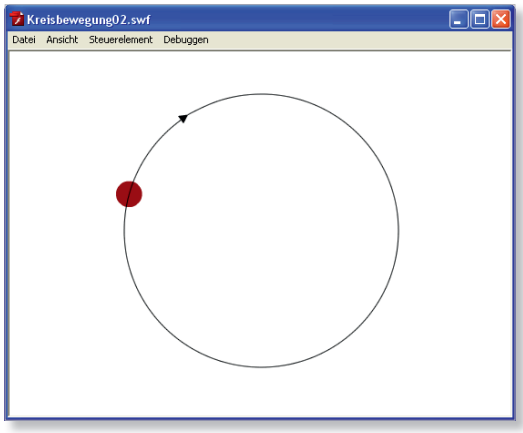
4 Kreisbewegung initiieren

Ergänzen Sie den Code um folgende Zeilen:

```
kreis_mc.addEventListener(Event.ENTER_FRAME,
moveBall);
function moveBall(e:Event):void {
   kreis_mc.x = startX+Math.cos(winkel)*radiusX;
   kreis_mc.y = startY+Math.sin(winkel)*radiusY;
   winkel += 0.1;
};
```

5 Film testen

Testen Sie den Film über Strg/⌘+↵.



Ergebnis der Übung:
09\Kreis_Elliptische_Bewegung\Kreisbewegung02.fla

◀ Abbildung 9.34
Der Bewegungsbereich des Movieclips

Schritt für Schritt: Spiralenförmige Bewegung

In diesem Workshop lernen Sie, wie Sie eine spiralenförmige Bewegung eines Objekts erzielen.

09\Kreis_Elliptische_Bewegung\Spiralenbewegung01.fla

1 Film öffnen

Öffnen Sie den Flash-Film *Spiralenbewegung01.fla* aus dem Ordner *Kreis_Elliptische_Bewegung*. Ausgangsbasis ist eine geskriptete kreisförmige Bewegung.

2 Code ergänzen

Um aus der kreisförmigen Bewegung eine spiralenförmige zu machen, müssen Sie den Radius über die Zeit verkleinern. Fügen Sie dem Code die folgenden fettgedruckten Codezeilen hinzu:

```
function moveBall(e:Event):void {

...
if (radiusX<1) {
      kreis_mc.removeEventListener(Event.ENTER_
FRAME,moveBall);
}
radiusX--;
radiusY--;
}
```

3 Film testen

Testen Sie den Film über Strg/⌘+↵. Sowohl der Radius der Bewegung auf der x-Achse als auch der Radius der Bewegung

Ergebnis der Übung:
09\Kreis_Elliptische_Bewegung\Spiralenbewegung02.fla

auf der y-Achse werden je Bild verkleinert. Sobald der x-Radius kleiner als 1 ist, wird die `ENTER_FRAME`-Ereignisprozedur entfernt, die Bewegung wird also gestoppt.

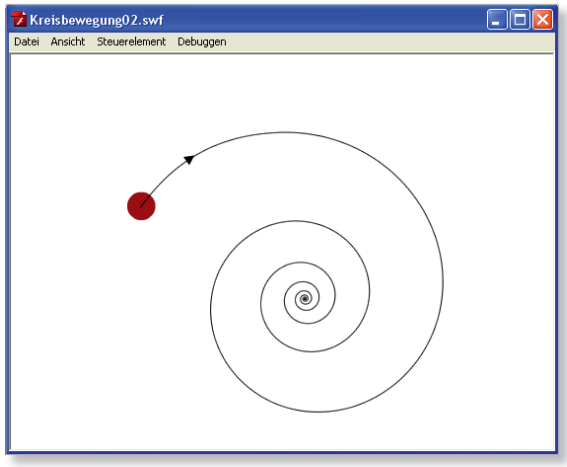


Abbildung 9.35 ▶
Der Bewegungsbereich des Movieclips

9.6.7 Winkel zwischen zwei Punkten berechnen

Gelegentlich möchte man den Winkel zwischen zwei Punkten ermitteln, z. B. um ein Objekt in die Richtung eines anderen Objekts zu drehen. Um den Winkel in Bogenmaß zwischen zwei Punkten zu berechnen, können Sie die `Math`-Methode `atan2` nutzen. Zur Berechnung des Winkels zwischen zwei Punkten wird zunächst die Distanz auf der x- und auf der y-Achse ermittelt:

```
var difX:Number = stage.mouseX - mc.x;
var difY:Number = stage.mouseY - mc.y;
```

Math.atan2(y,x)
Beachten Sie die unübliche Reihenfolge der Parameter. Zuerst wird die y-Distanz, dann die Distanz auf der x-Achse angegeben.

Über die Methode `atan2` der `Math`-Klasse lässt sich dann der Winkel im Bogenmaß (Radiant) wie folgt ermitteln:

```
var radiant:Number = Math.atan2(difY, difX);
```

09\Rotation\Rotation01.fla

Schritt für Schritt: Movieclip in Mausrichtung drehen

Der Workshop zeigt, wie Sie einen Movieclip in Mausrichtung drehen.

1 Film öffnen

Öffnen Sie den Flash-Film *Rotation01.fla* aus dem Ordner *Rotation*. Ausgangsbasis ist ein mittig registrierter Movieclip mit dem Instanznamen »car_mc«.

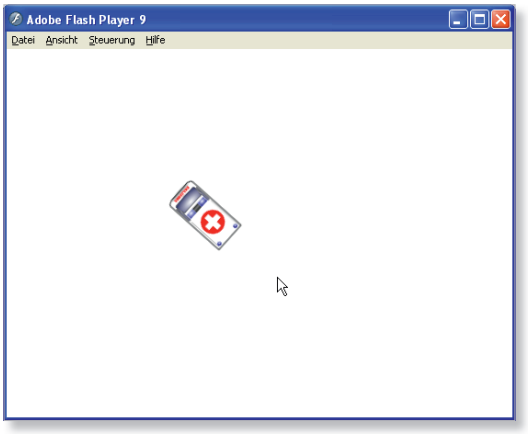
2 Code zuweisen

Weisen Sie dem ersten Schlüsselbild der Ebene »Actions« folgenden Code zu:

```
stage.addEventListener(MouseEvent.MOUSE_MOVE,
arrangeCar);
function arrangeCar(e:MouseEvent):void {
   var difX:Number = stage.mouseX-car_mc.x;
   var difY:Number = stage.mouseY-car_mc.y;
   var radiant:Number = Math.atan2(difY,difX);
   var grad:Number = radiant*(180/Math.PI);
   car_mc.rotation = grad;
   e.updateAfterEvent();
}
```

3 Film testen

Testen Sie den Film mit Strg/⌘+↵.



Ergebnis der Übung:
09\Rotation\Rotation02.fla

◂ Abbildung 9.36
Der Movieclip »car_mc« dreht sich automatisch in die Richtung des Mauszeigers.

Schritt für Schritt: Movieclip in Mausrichtung bewegen

09\Rotation\Rotation02.fla

Sie lernen in diesem Workshop, wie Sie einen Movieclip in die Richtung des Mauszeigers bewegen.

1 Film öffnen

Öffnen Sie den Flash-Film *Rotation02.fla* aus dem Ordner *Rotation*. Sobald der User klickt, soll sich der Movieclip »car_mc« in die Mausrichtung bewegen.

2 Zielkoordinaten festlegen

Ergänzen Sie den Code im ersten Schlüsselbild auf der Ebene »Actions« hinter dem bereits vorhanden Code um folgende Zeilen:

```
var targetX:Number;
var targetY:Number;
stage.addEventListener(MouseEvent.MOUSE_DOWN,
initCarMove);
function initCarMove(e:MouseEvent):void {
   targetX = stage.mouseX;
   targetY = stage.mouseY;
   car_mc.addEventListener(Event.ENTER_FRAME,moveCar);
}
```

Damit Sie auch innerhalb einer anderen Funktion auf die Werte der Variablen `targetX` und `targetY` zugreifen können, werden diese zunächst außerhalb der folgenden Ereignisprozedur definiert. Wenn die Maustaste gedrückt wird, werden den Variablen `targetX` und `targetY` die Zielkoordinaten zugewiesen. Anschließend wird die Ereignisprozedur `moveCar` aufgerufen.

3 Bewegung initiieren

Ergänzen Sie den Code um folgende Zeilen:

```
function moveCar(e:Event):void {
   var difX:Number = targetX-car_mc.x;
   var difY:Number = targetY-car_mc.y;
      car_mc.x += difX/4;
      car_mc.y += difY/4;
      if (Math.abs(difX)<1 && Math.abs(difY)<1) {
         trace("Car reached position.");
         car_mc.removeEventListener(Event.ENTER_
         FRAME,moveCar);
      }
}
```

Der Movieclip wird über eine `ENTER_FRAME`-Ereignisprozedur mittels eines einfachen Easings in Zielrichtung bewegt. Sobald der Abstand zwischen Movieclip und Ziel kleiner als 1 ist, wird die Ereignisprozedur entfernt, und eine Meldung wird im `Ausgabe`-Fenster ausgegeben. Beachten Sie dabei, dass die Werte `difX` und `difY` auch negativ sein können. Sie werden deshalb vorher über `Math.abs()` in absolute Werte umgewandelt.


09\Rotation\Rotation03.fla

4 Film testen

Testen Sie den Film über Strg/⌘+↵. ■

Methode	Beispiel	Beschreibung
abs	`Math.abs(-2.1);` Ergebnis: 2	Berechnet den absoluten Wert.
acos	`Math.acos(-1);` Ergebnis: 3.141...	Arkuskosinus (in Radiant)
asin	`Math.asin(-1);` Ergebnis: -1.570...	Arkussinus (in Radiant)
atan	`Math.atan(1);` Ergebnis: 0.78539	Arkustangens
atan2	`Math.atan2(10,0);` Ergebnis: 1.570...	Berechnet den Winkel des Punktes y/x im Bogenmaß.
ceil	`Math.ceil(3.4);` Ergebnis: 4	Rundet die Zahl auf.
cos	`Math.cos(Math.PI/2));` Ergebnis: 6.123...	Berechnet den Kosinus (in Radiant) des angegebenen Winkels.
exp	`Math.exp(2);` Ergebnis: 7.380...	Berechnet die Basis des natürlichen Logarithmus des angegebenen Exponenten.
floor	`Math.floor(3.7);` Ergebnis: 3	Rundet die Zahl ab.
log	`Math.log(2);` Ergebnis: 0.693...	Berechnet den natürlichen Logarithmus einer Zahl.
max	`Math.max(10,20);` Ergebnis: 20	Wertet zwei oder mehr Werte aus und gibt den höheren Wert zurück.
min	`Math.min(10,20);` Ergebnis: 10	Wertet zwei oder mehr Werte aus und gibt den niedrigeren Wert zurück.
pow	`Math.pow(2,3);` Ergebnis: 8	Berechnet x hoch y.
random	`Math.random();` Ergebnis: Zwischen 0 und 1.	Zufallszahl zwischen 0 und 1

round	`Math.round(2.4);`
	Ergebnis: 2
	`Math.round(2.5);`
	Ergebnis: 3
		Rundet den Wert auf die nächstliegende Zahl auf oder ab.

sin	`Math.sin(Math.PI/2);` Ergebnis: 1	Berechnet den Sinus des angegebenen Winkels.
sqrt	`Math.sqrt(4);` Ergebnis: 2	Berechnet die Quadratwurzel.
tan	`Math.tan(Math.PI/2));` Ergebnis: 16331778728383844	Berechnet den Tangens des angegebenen Winkels.

▲ **Tabelle 9.2**
Die wichtigsten Methoden der `Math`-Klasse

Konstante	Beispiel	Beschreibung
PI	`trace(Math.PI);` Ergebnis: 3.141592653589793	ein gerunderter Wert der Kreiszahl Pi
E	trace(Math.E); Ergebnis: 2.718281828459045	Eine mathematische Konstante für die Basis des natürlichen Logarithmus.
LN10	`trace(Math.LN10);` `Ergebnis: 2.302585092994046`	Eine mathematische Konstante für den natürlichen Logarithmus von 10.
LN2	`trace(Math.LN2);` `Ergebnis: 0.6931471805599453`	Eine mathematische Konstante für den natürlichen Logarithmus von 2.
LOG10E	`trace(Math.LOG10E);` `Ergebnis: 0.4342944819032518`	Eine mathematische Konstante für den Logarithmus zur Basis 10 der Konstante e (Math.E).
LOG2E	`trace(Math.LOG2E);` `Ergebnis: 1.4426950408889634`	Eine mathematische Konstante für den Zweierlogarithmus der Konstante e.
SQRT1_2	`trace(Math.SQRT1_2);` `Ergebnis: 0.7071067811865476`	Eine mathematische Konstante für die Quadratwurzel von ½.
SQRT2	`trace(Math.SQRT2);` `Ergebnis: 1.4142135623730951`	Eine mathematische Konstante für die Quadratwurzel von 2.

▲ **Tabelle 9.3**
Die wichtigsten Konstanten der `Math`-Klasse

9.7 Tween-Klassen

Es ist nicht immer sinnvoll, Animationen mit Hilfe von selbstgeschriebenen Funktionen zu erstellen. Wenn Sie beispielsweise sehr viele unterschiedliche Objekte auf unterschiedliche Weise animieren möchten, sind eigene Funktionen meist sehr aufwendig. Eine Alternative für Animationen in ActionScript bieten sogenannte Tween-Engines bzw. Tween-Klassen. In der Praxis werden aktuell noch sehr viele unterschiedliche Tween-Klassen verwendet. Das mag sich ändern, wenn sich die ein oder andere Klasse durchgesetzt hat und/oder eventuell von Adobe übernommen wird. Die Auswahl einer Tween-Engine ist aktuell einerseits Geschmackssache, hängt jedoch andererseits auch davon ab, welche Anforderungen Sie für Ihre Animationen an eine Tween-Engine stellen.

Performance-Beispiel auf DVD

Auf der DVD finden Sie unter *09\Adobe_Tween_Bug\beispiel.fla* ein Beispiel, das sowohl die schlechte Performance als auch den angesprochenen Ereignis-Fehler zeigt. Wie Sie sehen, bleiben einige Schneeflocken nach einiger Zeit einfach stehen, und die Ereignisprozedur `motionStarted` wird nicht aufgerufen.

9.7.1 Adobes Tween-Klasse

Adobe bietet seit ActionScript 2 eine eigene Tween-Klasse an. In ActionScript 3 können Sie die Klasse `Tween`, die zum Paket `fl.transitions` gehört, verwenden. Kurz gesagt, wenn Sie eine zuverlässige Tween-Klasse benötigen, die auch mit mehreren gleichzeitig animierten Objekten funktionieren soll, sollten Sie um die `Tween`-Klasse von Adobe einen Bogen machen. Die `Tween`-Klasse von Adobe bietet nur sehr rudimentäre Möglichkeiten. Darüber hinaus gibt es Performance-Probleme, wenn Sie

beispielsweise 50 Objekte nahezu gleichzeitig animieren möchten. Fast nebensächlich, dennoch erwähnenswert ist ein Fehler, der im Zusammenhang mit dem Ereigniss `TweenEvent.MOTION_START` auftritt: Die Ereignisprozedur, die über einen entsprechenden Ereignis-Listener definiert wird, wird nicht aufgerufen, was das Ereignis obsolet macht.

9.7.2 Tween-Engines

Glücklicherweise gibt es mehrere unterschiedliche freie Tween-Engines von Drittanbietern, die sich bewährt haben und von vielen Entwicklern in der Praxis verwendet werden. In der folgenden Tabelle finden Sie einen Auszug aktueller Tween-Engines.

Tween-Engine	Eigenschaften	Quelle
Tweener	Tweener ist eine solide Tween-Engine, die es u. a. ermöglicht, auf einfachste Weise mehrere Eigenschaften zu animieren. Auch Bitmap-Filter lassen sich mit Tweener animieren.	*http://code.google.com/p/tweener/*
TweenLite	TweenLite ist eine sehr leistungsfähige und kleine (3 KB) Tween-Engine, erhältlich für ActionScript 2 und 3.	*http://blog.greensock.com/tweenliteas3/*
TweenFilterLite	TweenFilterLite ist eine Ergänzung zu TweenLite, mit der sich auch Bitmap-Filter und Bildeffekte animieren lassen.	*http://blog.greensock.com/tweenfilterliteas3/*
TweenMax	TweenMax (11 KB) basiert auf TweenLite, bietet allerdings noch weitere Möglichkeiten, wie z. B. Bézier-Tweenings und einen Sequenzer.	*http://blog.greensock.com/tweenmaxas3/*
GTween	GTween wird von Grant Skinner entwickelt und bietet eine leichte und schnelle Tween-Engine mit sehr vielen Einstellungsmöglichkeiten.	*http://www.gskinner.com/blog/archives/2008/11/gtween_beta_3_a.html*
Tweensy	Tweensy ist zum Zeitpunkt der Drucklegung noch im Beta-Status und sieht auf den ersten Blick sehr vielsprechend aus. Die Tween-Engine wird als Open Source unter der MIT-Lizenz veröffentlicht und verspricht, eine leistungsstarke und schlanke Tween-Engine zu sein.	*http://code.google.com/p/tweensy/*

▲ **Tabelle 9.4**
Bewährte, frei verfügbare und gute Tween-Engines von Drittanbietern

9.7.3 TweenLite

Für die meisten Animationszwecke ist die sehr kleine und leistungsfähige TweenLite-Engine von Jack Doyle ausreichend. Die Anwendung der Engine wird auf den folgenden Seiten erläutert. Kopieren Sie zunächst das Verzeichnis *gs* der gewünschten ActionScript Version, in diesem Fall das Verzeichnis *gs* aus dem

Version
Zum Zeitpunkt der Drucklegung war die Version 10.09 aktuell. Auf der beiligenden DVD finden Sie sowohl eine Version für ActionScript 2 als auch für ActionScript 3 unter *Software\TweenLite*. Bei neueren Versionen sind von den hier dargestellten Erläuterungen abweichende Funktionen nicht ausgeschlossen.

Verzeichnis *AS3*, in ein beliebiges Verzeichnis. Auf der DVD finden Sie die TweenLite-Engine unter *Software\TweenLite*.

Erstellen Sie einen neuen Flash-Film, und speichern Sie den Film ebenfalls in das Verzeichnis. Anschließend importieren Sie die Paket über folgende Zeile:

```
import gs.TweenLite;
```

Methoden der TweenLite-Klasse | Sie können die Klasse `TweenLite` auf unterschiedliche Weise verwenden. Einerseits können Sie statische Methoden der Klasse benutzen. In diesem Fall müssen Sie kein Objekt der Klasse initialisieren. Das hat den Vorteil, dass Sie sich um die Entfernung des Objekts aus dem Speicher nicht kümmern müssen. Der folgende Code führt dazu, dass ein Movieclip mit dem Instanznamen »mc« innerhalb von 1 Sekunde auf die x-Koordinate 300 bewegt wird.

```
TweenLite.to(mc, 1, {x: 300});
```

Nutzungsrechte
TweenLite kann für private als auch, eingeschränkt, für kommerzielle Anwendungen genutzt werden. Weitere Informationen zu Nutzungsrechten finden Sie in der beiligenden *readme.txt*.

Diese Methode wird im Folgenden bevorzugt verwendet. Andererseits können Sie auch ein Objekt der `TweenLite`-Klasse initialisieren. Das hat den Vorteil, dass Sie so mehrere Tweens erstellen können und das Tween einen Bezeichner besitzt, den Sie zur Steuerung oder Entfernung des Objekts verwenden können. Das gleiche Beispiel mit einem explizit definierten `TweenLite`-Objekt:

```
var myTween:TweenLite = new TweenLite(mc, 1, {x:300});
```

Die Klasse verfügt über weitere Methoden, die in der folgenden Tabelle aufgelistet sind.

Methode	Beispiel	Beschreibung
`from`	`import gs.TweenLite;` `TweenLite.from(mc,1,{x:500});`	Tweent die Werte der Eigenschaften des Objekts von den angegebenen Werten zu den Werten, die es im Moment besitzt.

`delayedCall`	`import gs.TweenLite;`
	`TweenLite.to(mc,1,{x:500});`
	`TweenLite.delayedCall(1, moveDone, [mc])`
	`function moveDone(target:MovieClip):void {`
	`  trace(target +" wurde bewegt.");`
	`}`
		Ruft die angegebene Funktion nach `x` Sekunden auf. Dabei können beliebige Argumente als Array an die Funktion übergeben werden.

▲ **Tabelle 9.5**
Methoden der `TweenLite`-Klasse

Methode	Beispiel	Beschreibung

killTweensOf	`import gs.TweenLite;`
	`TweenLite.to(mc,1,{x:500});`
	`TweenLite.killTweensOf(mc,true);`
		Entfernt unverzüglich alle Tweens des angegebenen Objekts. Dabei können Sie angeben, ob das Objekt unverzüglich auf die Zielwerte gesetzt werden soll (`true`) oder nicht (`false`).

killDelayedCallsTo	`…`
	`TweenLite.delayedCall(1, moveDone, [mc])`
	`function moveDone(target:MovieClip):void {`
	`  trace(target +" wurde bewegt.");`
	`}`
	`TweenLite.killDelayedCallsTo(moveDone);`
		Entfernt alle Aufrufe einer Funktion, die über die Methode `delayedCall` eingerichtet wurde.

removeTween	`import gs.TweenLite;`
	`var myTween:TweenLite = new TweenLite(mc,`
	`1, {x:300});`
	`TweenLite.removeTween(myTween);`
		Entfernt eine Tween-Instanz.

▲ **Tabelle 9.5**
Methoden der `TweenLite`-Klasse (Forts.)

Mehrere Eigenschaften und Sequenzen animieren | Natürlich können Sie auch mehrere Eigenschaften gleichzeitig animieren. In dem folgenden Beispiel wird der Movieclip mit dem Instanznamen »mc« auf die x-Koordinate 300 bewegt und zeitgleich ausgeblendet:

```
TweenLite.to(mc, 1, {x: 300,alpha:0});
```

Sequenzierte Animationen, d. h. Animationen, die Sie in mehrere Sequenzen aufteilen möchten, sind ebenso möglich. Dazu folgendes Beispiel:

```
TweenLite.to(mc, 1, {y: 200});
TweenLite.to(mc,3,{alpha:0,delay:1,overwrite:0});
```

Zunächst wird der Movieclip auf die y-Koordinate 200 verschoben. Nachdem er die Position erreicht hat, wird er ausgeblendet. Über die Eigenschaft `delay` geben Sie die Verzögerung (in Sekunden) an. Da die erste Sequenz 1 Sekunde benötigt, wird bei der zweiten Sequenz eine Verzögerung von einer Sekunde eingestellt. Standardmäßig werden Tweens überschrieben. Das bedeutet, dass ein Tween, das aktuell läuft, überschrieben wird, sobald ein neues Tween über die Methode `to` ausgeführt wird. Um das zu vermeiden, setzen wir hier die Eigenschaft `overwrite` auf `0`. Es gibt noch weitere gültige Werte für die Eigenschaft, die

in der folgenden Tabelle aufgelistet sind. Beachten Sie, dass in der Lite-Version standardmäßig ausschließlich die Werte 0 und 1 eingestellt werden können.

OverwriteManager
Wenn Sie den OverwriteManager verwenden, wird der Flash-Film ca. 1 Kilobyte größer.

Um auch die anderen Werte in der Lite-Version verwenden zu können, müssen Sie zusätzlich die Klasse `OverwriteManager` importieren und durch Aufruf der Klassen-Methode `init` initialisieren. Anschließend können Sie auch die anderen Werte verwenden.

```
import gs.OverwriteManager;
OverwriteManager.init();
```

Overwrite-Eigenschaftswert	Beispiel/Beispiel-Beschreibung	Beschreibung
0 (NONE)	`TweenLite.to(mc, 1, {y: 200});` `TweenLite.to(mc,3,{y:0,delay: 1, overwrite:0});`	Kein Tween wird überschrieben – die schnellste Methode. Sie müssen jedoch selbst darauf achten, dass keine überlappenden Eigenschaften eines Objekts animiert werden.
1 (ALL)	`TweenLite.to(mc, 1, {x: -100});` `TweenLite.to(mc,1,{x:550, overwrite:1});` Der Movieclip bewegt sich sofort zu x:550.	Alle Tweens eines Objekts werden sofort überschrieben.
2 (AUTO)	`TweenLite.to(mc, 1, {x: -100,y:200});` `TweenLite.to(mc,1,{x:550, overwrite:2});` Nur die x-Position des ersten Tweens wird überschrieben.	Sucht und überschreibt nur individuelle Eigenschaften, die sich überlappen.
3 (CONCURRENT)	`TweenLite.to(mc, 1, {x: -100,y:200});` `TweenLite.to(mc,1,{x:550, overwrite:3});` Das erste Tween wird vom zweiten überschrieben. Sollte im zweiten eine Verzögerung (»delay«) von 1 Sekunde eingestellt werden, würde der erste nicht überschrieben.	Überschreibt alle aktiven Tweens eines Objekts, wenn das Tween gestartet wird.

▲ **Tabelle 9.6**
Eigenschaftswerte für die `overwrite`-Eigenschaft

Easing-Equations von Robert Penner

Die mitgelieferten Easing-Funktionen wurden von Robert Penner entwickelt und unter der Open-Source-Lizenz BSD veröffentlicht. Für weitere Informationen zu Nutzungsrechten lesen Sie die *easing_readme.txt* im Verzeichnis *TweenLite/gs/easing*.

Easing mit TweenLite | Wie fast jede aktuell verwendete Tween-Engine bietet auch TweenLite Easing an. Um ein Tween mit einer Easing-Gleichung zu versehen, müssen Sie zunächst die mitgelieferten Klassen importieren. Dazu dient folgende Zeile.

```
import gs.easing.*;
```

Anschließend können Sie die Eigenschaft `ease` mit einer der verfügbaren Easing-Funktionen definieren.

```
TweenLite.to(mc, 3, {scaleX:"2", ease:Quad.easeOut});
```

Folgende Easing-Typen stehen Ihnen zur Verfügung.

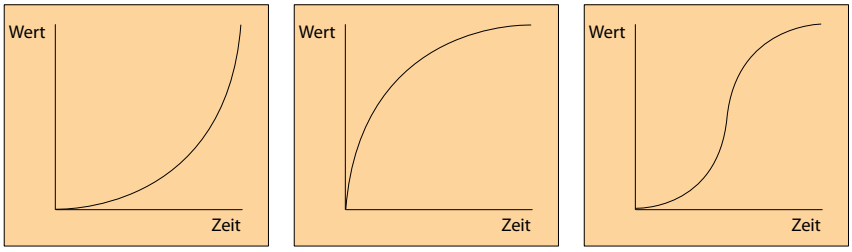
- `Back`
- `Bounce`
- `Circ`
- `Cubic`
- `Elastic`
- `Expo`
- `Linear`
- `Quad`
- `Quart`
- `Quint`
- `Sine`
- `Strong`

Bis auf den Typ `Linear` unterstützen alle Typen folgende Easing-Methoden.

- `easeIn`: Die Werte erhöhen sich am Anfang nur wenig und mit zunehmender Dauer dann mehr (beschleunigte Bewegung).
- `easeOut`: Die Werte erhöhen sich am Anfang sehr stark und mit zunehmender Dauer dann weniger (abbremsende Bewegung).
- `easeInOut`: Die Werte erhöhen sich sowohl am Anfang wenig als auch dann am Ende wieder wenig (beschleunigte und abgebremste Bewegung).

Typ: Linear

Bei dem Typ `Linear` stehen Ihnen genau genommen vier Methoden zur Auswahl, die jedoch keinen Einfluß auf die Bewegung haben. Es handelt sich dabei um eine lineare Bewegung. Der Easing-Typ ist damit überflüssig.



▲ **Abbildung 9.37**
Links: `Circ.easeIn`, Mitte: `Circ.easeOut`, Rechts: `Circ.easeInOut`

Tween-Ereignisse | An der `TweenLite`-Klasse können keine Ereignis-Listener registriert werden. Sie können jedoch über spezielle Eigenschaften Funktionen angeben, die aufgerufen werden, wenn ein Tween startet, läuft und beendet ist. Dazu dienen die Eigenschaften `onStart`, `onUpdate` und `onComplete` vom Datentyp Function. Dazu folgendes Beispiel.

```
import gs.TweenLite;
TweenLite.to(mc,1,{rotationY:360,onStart:hasStarted,
onUpdate:isTweening,onComplete:hasFinished});
function hasStarted():void {
   trace("Tween wurde gestartet.");
}
function isTweening():void {
   trace("Tween läuft.");
}
function hasFinished():void {
   trace("Tween wurde beendet.");
}
```

Optional können Sie über die Eigenschaften `onStartParams`, `onUpdateParams` und `onCompleteParams` vom Datentyp Array eigene Argumente an die Funktionen übergeben. Dazu folgendes Beispiel für die Eigenschaft `onStartParams`:

```
import gs.TweenLite;
TweenLite.to(mc,1,{rotationY:360,onStart:hasStarted,on
StartParams:[mc,"Startmeldung:"]});
function hasStarted(target:MovieClip,msg:String):void {
   trace(msg+target);
}
```

Schritt für Schritt: 3D-Flip mit TweenLite

In diesem Workshop lernen Sie, wie Sie TweenLite einsetzen, um einen Movieclip auf der y-Achse zu rotieren und so einen 3D-Effekt (Flip) zu erzielen..

09\FlipPlane\flipPlane_01.fla

1 Flash-Film öffnen

Öffnen Sie den Flash-Film *flipPlane_01.fla* aus dem Verzeichnis *FlipPlane*. Auf der Hauptzeitleiste liegen zwei Movieclips mit den Instanznamen »plane_mc« und »flip_mc«. Der Movieclip »flip_mc« dient als Schaltfläche. Wählen Sie den Movieclip »plane_mc«, aus und wechseln Sie mit Strg/⌘+E in den Bearbeitungsmodus. Auf der Ebene »Rahmen« ist ein Rechteck ohne Füllung eingezeichnet. Die darunterliegenden Ebenen »content0« und »content1« enthalten jeweils einen Movieclip mit dem Instanznamen »content0_mc« bzw. »content1_mc« mit jeweils einem Bitmap. Beide Movieclips sind mittig innerhalb des Movieclips »plane_mc« positioniert. Wechseln Sie zurück zur Hauptzeitleiste.

2 Movieclip rotieren

Wählen Sie das erste Schlüsselbild der Ebene »Actions« aus, und fügen Sie zunächst folgenden Code ein:

```
1:   import gs.TweenLite;
2:   flip_mc.addEventListener(MouseEvent.CLICK,
     rotatePlane);
3:   plane_mc.content1_mc.visible=false;
4:   var targetRotation:Number=180;
5:   var planeWidth:Number=plane_mc.width;
6:   function rotatePlane(e:MouseEvent):void {
7:      TweenLite.to(plane_mc,1,{rotationY:
       targetRotation,onUpdate:checkRotation,
       onComplete:changeDirection});
8:   }
```

Zunächst wird in Zeile 1 die `TweenLite`-Klasse importiert. In Zeile 2 wird ein Ereignis-Listener am Objekt »flip_mc« registriert, der die Ereignisprozedur `rotatePlane` aufruft, wenn der Movieclip angeklickt wird. Der Inhalt des Movieclips »content1_mc«, der im Movieclip »plane_mc« liegt wird zu Beginn in Zeile 3 ausgeblendet. Zunächst soll sich die Fläche samt Inhalt auf der y-Achse um 180 Grad drehen. Dazu wird in Zeile 4 eine entsprechende Variable definiert. Die Breite des Movieclips »plane_mc« wird der Variablen `planeWidth` zugewiesen. Sie wird später benötigt, um das gespiegelte Bild zu positionieren.

Klickt der Benutzer auf den Button, wird der Movieclip »plane_mc« auf der y-Achse zunächst auf 180 Grad gedreht. Während das Tween ausgeführt wird, wird die Funktion `checkRotation` mehrmalig aufgerufen. Sobald das Tween abgeschlossen ist, wird die Funktion `changeDirection` aufgerufen.

3 Inhalte ein- bzw. ausblenden und Movieclip positionieren

Ergänzen Sie den Code um folgende Zeilen:

```
1:   function checkRotation():void {
2:      if (plane_mc.rotationY>=90 && targetRotation==
        180) {
3:         plane_mc.content1_mc.scaleX=-1;
4:         plane_mc.content1_mc.x=planeWidth/2;
5:         plane_mc.content0_mc.visible=false;
6:         plane_mc.content1_mc.visible=true;
```

```
7:       } else if (plane_mc.rotationY >= 270 &&
         targetRotation == 360) {
8:          plane_mc.content1_mc.scaleX=1;
9:          plane_mc.content0_mc.visible=true;
10:          plane_mc.content1_mc.visible=false;
11:       }
12:    }
```

Sobald die Rotation auf der y-Achse 90 Grad überschreitet und das Ziel der Drehung 180 Grad ist, wird der Movieclip »content0_mc« aus- und der Movieclip »content1_mc« eingeblendet (Zeile 5, 6). Zusätzlich wird in Zeile 3 und 4 der Movieclip »content1_mc« auf der x-Achse auf –1 skaliert (gespiegelt) und neu positioniert.

Klickt der Benutzer nach einer halben Drehung wieder auf den Button, wird die Fläche von 180 Grad auf 360 Grad gedreht. In diesem Fall wird der Movieclip »content0_mc« aus- (Zeile 9) und der Movieclip »content1_mc« wieder eingeblendet (Zeile 10), wenn die Rotation in y-Richtung den Wert 270 überschreitet. Zusätzlich wird die Skalierung in x-Richtung des Movieclips »content1_mc« in Zeile 8 wieder auf 1 zurückgesetzt.

4 Den Zielwert für die Rotation ändern

Ergänzen Sie den Code um folgende Zeilen.

```
1:   function changeDirection():void {
2:      if (targetRotation==180) {
3:         targetRotation=360;
4:      } else {
5:         plane_mc.rotationY=0;
6:         targetRotation=180;
7:      }
8:   }
```

Sobald eine Rotation abgeschlossen ist, wird die Funktion `changeDirection` aufgerufen. Falls der bisherige Zielwert für die Rotation 180 beträgt, wird der Wert auf 360 geändert (Zeile 2, 3). Sollte der bisherige Zielwert einen anderen Wert (360) besitzen, wird der Zielwert auf 180 gesetzt (Zeile 6). Zusätzlich wird die Rotation des Movieclips »plane_mc« in y-Richtung dann auf 0 gesetzt. Tatsächlich besitzt der Movieclip zu diesem Zeitpunkt eine y-Rotation von 360 Grad. Der Wert muss auf 0 zurückgesetzt werden, damit die `if`-Bedingung in der Funktion `checkRotation` weiterhin funktioniert.

5 **Fertig! Flash-Film testen.**
Testen Sie den Flash-Film mit Strg/⌘+↵. ■

Ergebnis der Übung:
09\FlipPlane_02.fla

Weitere Eigenschaften eines Tweens | Jedem Tween können Sie weitere besondere Eigenschaften zuweisen, die in der folgenden Tabelle aufgelistet sind.

▼ **Tabelle 9.7**
Die wichtigsten zusätzlichen Eigenschaften eines Tweens

Eigenschaft	Datentyp	Beispiel	Beschreibung

`autoAlpha`	`Number`	`import gs.TweenLite;`
		`TweenLite.to(mc,1,{autoAlpha:0});`
		`TweenLite.to(mc,1,{delay:1,`
		`autoAlpha:1,overwrite:0});`
			Ähnlich wie die Eigenschaft `alpha`, mit dem Unterschied, dass ein Objekt, dessen Alphawert 0 erreicht hat, durch Setzen der Eigenschaft `visible` auf false tatsächlich inaktiviert wird. Analog dazu wird der Wert der Eigenschaft `visible` auf `true` gesetzt, wenn das Objekt einen größeren Alphawert als 0 erhält. Ein Objekt, dessen Eigenschaft auf `visible=false` gesetzt ist, reagiert nicht mehr auf Maus- oder Tastaturereignisse.

`visible:`
`Boolean`
	`Boolean`	`import gs.TweenLite;`
		`TweenLite.to(mc,1,{alpha:0.1,`
		`visible:false});`
			Setzt die `visible`-Eigenschaft eines Objektes am Ende eines Tweens auf den angegebenen Wert.

`volume`	`Number`	`var sndLoop:SoundLoop = new`
		`SoundLoop();`
		`var soundChannel:SoundChannel;`
		`soundChannel = sndLoop.`
		`play(0,1000);`
		`import gs.TweenLite;`
		`TweenLite.to(soundChannel,5,`
		`{volume:0.1});`
			Tweent die Lautstärke eines Objektes mit einer `SoundTransform`-Eigenschaft (Movieclip/SoundChannel/NetStream etc.) Siehe dazu auch Kapitel 14, »Sound«.

`tint`	`uint`	`import gs.TweenLite;`
		`TweenLite.`
		`to(mc,1,{tint:0x99CC00});`
			Färbt das Objekt mit der angegebenen Farbe.

`removeTint`	`Boolean`	`import gs.TweenLite;`
		`TweenLite.`
		`to(mc,1,{tint:0x99CC00});`
		`TweenLite.to(mc,1,{delay:2,`
		`overwrite:0,removeTint:true});`
			Entfernt eine Färbung des Objekts, wenn der Wert der Eigenschaft auf `true` gesetzt wird.

`frame`	`int`	`import gs.TweenLite;`
		`TweenLite.to(mc,5,{delay:`
		`2,tint:0x99CC00,frame:50});`
			Bewegt den Abspielkopf eines Movieclips Bild für Bild bis hin zum angegebenen Bild der Zeitleiste des Movieclips.

`persists`	`Boolean`	`import gs.TweenLite;`
		`var myTween:TweenLite = new`
		`TweenLite(mc, 1,`
		`{x:300,persists:true});`
			Wird der Wert auf `true` gesetzt, wird eine `TweenLite`-Instanz nach der Beendigung des Tweens nicht automatisch vom GarbageCollector entfernt.

Eigenschaft	Datentyp	Beispiel	Beschreibung
renderOnStart	Boolean	import gs.TweenLite; TweenLite. from(mc,5,{alpha:0,delay: 2,renderOnStart:true}); TweenLite. from(mc2,5,{alpha:0,delay: 2}); (Tipp: Testen Sie das mit zwei Movieclip-Instanzen.)	Falls Sie die Methode from (die Eigenschaft runBackwards mit dem Wert true) mit einer Verzögerung verwenden und Sie verhindern wollen, dass das Tween gerendert wird, bevor es tatsächlich startet, setzen Sie diese Eigenschaft auf true.
runBackwards	Boolean	import gs.TweenLite; TweenLite.to(mc,1,{x:0, runBackwards:true});	Tauscht die Start- und Zielwerte eines Tweens aus (true). (Bei Verwendung der Methode from wird die Eigenschaft automatisch auf true gesetzt.)

▲ **Tabelle 9.7**
Die wichtigsten zusätzlichen Eigenschaften eines Tweens (Forts.)

Schritt für Schritt: Schneeflockensimulation mit TweenLite

In diesem Workshop lernen Sie, wie Sie mit der TweenLite-Klasse und einem Timer-Objekt eine Schneeflocken simulieren. Zusätzlich lernen Sie eine sinvolle Anwendung der Eigenschaft onCompleteParams der TweenLite-Klasse kennen.

09\Schneeflocken_Simulation\step_01.fla

1 Flash-Film öffnen

Öffnen Sie den Flash-Film *step_01.fla* im Verzeichnis *Schneeflocken_Simulation*. In der Bibliothek finden Sie einen Movieclip, dem die Klasse Snowflake zugewiesen wurde.

[!] Eigenschaftskonflikte
Es gibt einige Eigenschaften, bei denen Konflikte auftreten können. Wenn Sie beispielsweise die Eigenschaft autoAlpha verwenden, hat die zusätzliche Verwendung der Eigenschaft visible keinen Einfluss.

2 Timer-Objekt initialisieren

Wählen Sie das erste Schlüsselbild der Ebene »Actions« aus, und weisen Sie dem Schlüsselbild zunächst folgenden Code zu:

```
import gs.TweenLite;
var myTimer:Timer=new Timer(200,int.MAX_VALUE);
myTimer.addEventListener(TimerEvent.TIMER,initFlake);
myTimer.start();
```

Es wird ein Timer-Objekt initialisiert. Das Objekt sorgt dafür, dass die Ereignisprozedur initFlake mit einem zeitlichen Abstand von 200 Millisekunden aufgerufen wird.

3 Snowflake-Objekte erstellen, positionieren, skalieren und Animation starten

Ergänzen Sie den Code um folgende Zeilen.

```
1:   function initFlake(e:TimerEvent):void {
2:      trace("Anzahl Flakes: "+numChildren);
3:      var anzahl:uint = 3;
4:      for(var i:uint = 0;i<anzahl;i++) {
5:         var mySnowflake:Snowflake = new
           Snowflake();
6:         mySnowflake.x=randomExt(0,550);
7:         mySnowflake.y=-10;
8:         var destX:Number=randomExt(mySnowflake.
           x-250,mySnowflake.x+250);
9:         var destY:Number=stage.stageHeight+10;
10:         var scaling:Number=Math.random();
11:         var time:Number=randomExt(5,9);
12:         mySnowflake.scaleX=mySnowflake.
            scaleY=scaling;
13:         addChild(mySnowflake);
14:         TweenLite.to(mySnowflake,time,{x:destX,y:
            destY,onComplete:killFlake,onCompletePara
            ms:[mySnowflake]});
15:      }
16:   }
```

Ab Zeile 4 werden mit Hilfe einer `for`-Schleife drei Schneeflocken erstellt (Zeile 5), zufällig skaliert (Zeile 10, 12), positioniert (Zeile 6, 7) und in die Anzeigeliste (Zeile 13) eingefügt. Die Dauer eines Tweens wird ebenfalls zufällig gewählt (Zeile 11). Sollten Sie auf diese Weise mehrere Objekte erstellen, empfiehlt es sich, während der Laufzeit zu überprüfen, wie viele Objekte auf der Bühne aktuell platziert sind. Dazu können Sie die Eigenschaft `numChildren` nutzen, die angibt, wie viele Anzeigeobjekte sich in der Anzeigeliste befinden. Später werden Objekte, die außerhalb der Bühne liegen, wieder entfernt, so dass ein Volllaufen des Speichers verhindert wird. Sie sehen, dass das Löschen der Objekte funktioniert, da die Anzahl der Objekte auf der Bühne nach einiger Zeit stagniert. Beachten Sie, dass die Funktion `killFlake` aufgerufen wird, sobald ein Tween beendet wurde (Zeile 14). Damit das Objekt, das das Tween beendet hat, entfernt werden kann, wird es über die Eigenschaft `onCompleteParams` (Zeile 14) an die Funktion übergeben.

4 Zufallsfunktion definieren und Objekt nach dem Tween aus der Anzeigeliste entfernen

Ergänzen Sie den Code um folgende Zeilen:

```
function randomExt(minVal:Number,maxVal:Number):
Number {
   return Math.floor(Math.random() *
   (1+maxVal-minVal)) + minVal;
}
function killFlake(target:MovieClip):void {
   removeChild(target);
}
```

Die Funktion `randomExt` erwartet zwei Werte und gibt eine zufällige Zahl aus dem angegebenen Wertebereich zurück. Wenn Sie beispielsweise die Zahlen 0 und 10 an die Funktion übergeben, gibt die Funktion eine zufällige Zahl zwischen 0 und 10 zurück. Wie bereits erwähnt, wird die Funktion `killFlake` aufgerufen, wenn ein Tween beendet wurde. Über die Methode `removeChild` wird der Movieclip aus der Anzeigeliste entfernt.

Ergebnis der Übung:
09\Schneeflocken_Simulation\step_02.fla

5 Fertig! Flash-Film testen

Testen Sie den Flash-Film über Strg/⌘+↵. Ändern Sie den Wert der Variablen `anzahl` vor der `for`-Schleife, wenn Sie mehr oder weniger Flocken möchten. Probieren Sie ruhig auch andere Wertebereiche für die zufällige Skalierung und für die zufällige Animationsszeit aus. ■