

jetzt lerne ich

# ActionScript

Der einfache Einstieg in die  
Programmierung mit Flash MX/MX 2004

DIRK LOUIS



Markt+Technik

## Die Programmierumgebung

Sollten Sie im vorangehenden Kapitel den Eindruck gewonnen haben, dass die ActionScript-Programmierung fast ausschließlich im Bedienfeld AKTIONEN stattfindet, so hat Sie Ihr Eindruck nicht getäuscht. Das AKTIONEN-Bedienfeld ist der Dreh- und Angelpunkt der ActionScript-Programmierung, und der sichere Umgang mit dem Bedienfeld ist die Grundvoraussetzung für (relativ) frustrationsfreies Programmieren mit ActionScript. Aus diesem Grunde ist das folgende Kapitel dem AKTIONEN-Bedienfeld und den anderen Programmierwerkzeugen der Flash-Oberfläche gewidmet.

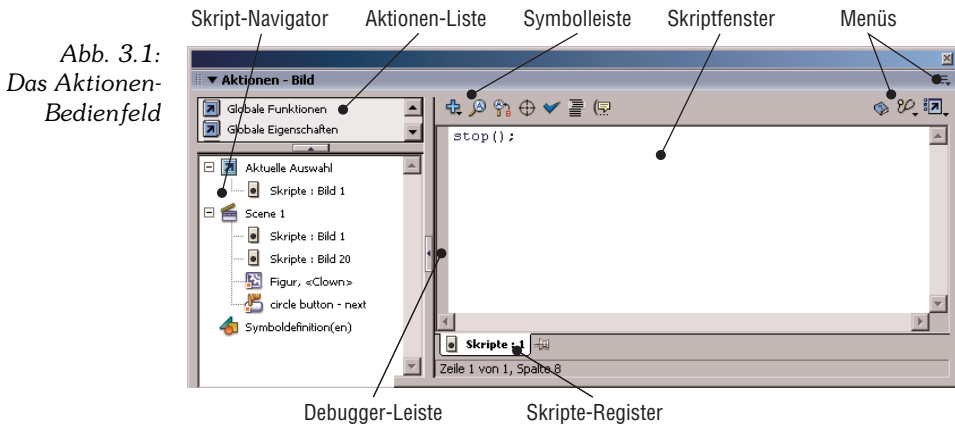
### Im Einzelnen lernen Sie in diesem Kapitel

- alles Wichtige über das AKTIONEN-Bedienfeld,
- wie man in ActionScript Daten zur Kontrolle ausgeben kann,
- wie man ActionScript-Code testen und debuggen kann,
- wer den ActionScript-Code eigentlich ausführt.

### 3.1 Das Aktionen-Bedienfeld

ActionScript-Code kann, wie Sie wissen, nur mit Schlüsselbildern, Schaltflächen-, Komponenten- und Filmsequenzinstanzen verbunden werden. Zuweisung und Bearbeitung des Codes erfolgen stets über das Bedienfeld AKTIONEN.

*Drücken Sie **F9**, um das AKTIONEN-Bedienfeld zu öffnen.*



### 3.1.1 Die einzelnen Teilbereiche

Sieht man von untergeordneten Dekorationen ab ist das *Aktionen*-Bedienfeld dreigeteilt: in zwei kleinere Fenster zur Linken und ein größeres Fenster zur Rechten (siehe Abbildung 3.1).

#### Das Skriptfenster (rechts)

Das rechte Fenster birgt den eigentlichen Editor. Hier tippen Sie Ihren Action-Script-Code ein.

Der Editor unterstützt Sie unter anderem mit:

- farblicher Hervorhebung unterschiedlicher Syntaxelemente
- automatischer Codeeintrückung
- Codehinweisen (kleine Fenster, die aufspringen, wenn Sie die öffnende Klammer hinter `on`, `onClipEvent` oder einem Funktionsnamen bzw. einen Punkt hinter einem Instanznamen eingeben (siehe Abbildung 3.2)

(Für Instanzen und Objekte werden Codehinweise nur angezeigt, wenn Typinformationen vorliegen. Für Objekte geben Sie den Typ bei der Erzeugung an, siehe Kapitel 9, für Instanzen von Filmsequenzen, Schaltflächen etc. können Sie dem Instanznamen ein spezielles Suffix anhängen, das den Typ angibt, siehe hierzu die Online-Hilfe unter [ACTIONSCRIPT-REFERENZHANDBUCH/SKRIPTS SCHREIBEN UND DEBUGGEN/ACTIONSCRIPT-EDITOR VERWENDEN/CODE ZUM AUSLÖSEN VON CODEHINWEISEN SCHREIBEN.](#))

- Zeilennummerierung

Die meisten dieser Optionen können Sie über den VOREINSTELLUNGEN-Dialog (Aufruf über **[Strg]+[U]**) konfigurieren. Die Zeilennummerierung schalten Sie über das ANSICHTSOPTIONEN-Menü des Bedienfelds ein.

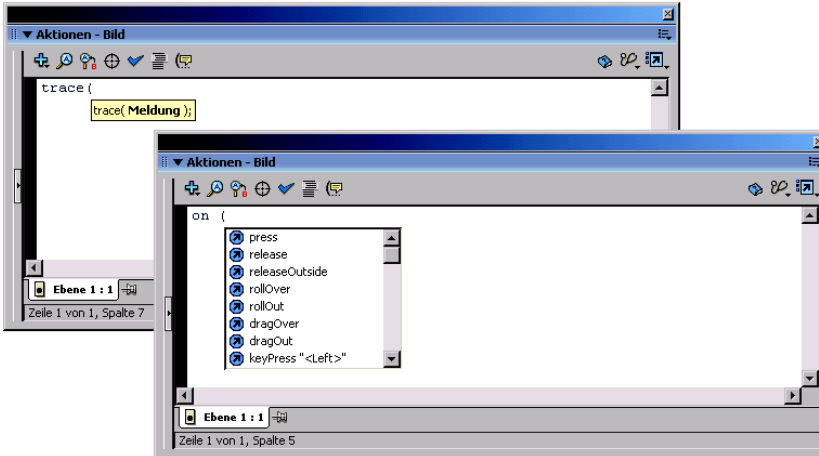


Abb. 3.2:  
Codehinweise  
im Aktionen-  
Bedienfeld

Im Grunde genommen würde dieses Teilfenster bereits ausreichen – die beiden linken Fenster dienen lediglich der Bequemlichkeit und besseren Übersicht.

### Der Skript-Navigator (links unten)

Dieses äußerst nützliche Fenster hilft Ihnen, die Übersicht über die Codeblöcke zu behalten, die mit den verschiedenen Elementen eines Flash-Films verbunden sind. Bilder werden unter Angabe der Ebene und der Bildnummer, Symbolinstanzen unter Angabe des Symbolnamens und – soweit vom Autor spezifiziert – dem Instanznamen aufgeführt.

Besondere Aufmerksamkeit verdient die AKTUELLE AUSWAHL. Unter diesem Knoten wird angezeigt, welches Element gerade in der Authoring-Umgebung (in diesem Fall Bühne oder Zeitleiste) ausgewählt ist. Der im Skriptfenster eingegebene oder angezeigte Code gehört immer zu dem aktuell ausgewählten Element!

Darunter werden nach Szenen geordnet diejenigen Filmelemente (Bilder, Filmsequenz-, Schaltflächen- und Komponenteninstanzen) aufgeführt, die mit ActionScript-Code verbunden sind. Wenn Sie eines dieser Elemente per Doppelklick auswählen, wird der mit dem Element verbundene Code in das Skriptfenster geladen.

Gibt es in der Filmbibliothek Filmsequenzsymbole, die ActionScript-Code enthalten (beispielsweise in den Schlüsselbildern der Filmsequenz) werden die Symbole unter der Rubrik `SYMBOLDEFINITION(EN)` aufgeführt.



Sollte Ihnen die Funktionalität des Skript-Navigators einmal nicht ausreichen, rufen Sie den Film-Explorer auf (Befehl `FENSTER/ANDERE BEDIENFELDER/FILM-EXPLORER`) Dieser präsentiert Ihnen die Elemente und Codeblöcke Ihres Films in einer hierarchischen, konfigurierbaren Ansicht und ermöglicht auch die Suche nach beliebigen Textstellen.

### Die Aktionen-Liste (links oben)

In diesem hierarchisch strukturierten Fenster können Sie ActionScript-Elemente (Operatoren, Funktionen, Klassen etc.) auswählen und per Doppelklick in den Code des Skriptfensters einfügen.

Um beispielsweise eine Ereignisbehandlungsdefinition für eine Filmsequenzinstanz zu beginnen, könnten Sie den Knoten  `Globale Funktionen / MovieClipsteuerung` erweitern, zur Aktion `onClipEvent` scrollen und diese doppelt anklicken. Danach können Sie den Code im rechten Fenster weiterbearbeiten. In der Praxis werden Sie diese Möglichkeit jedoch eher selten nutzen, da Sie den Code viel schneller selbst eingetippt haben.

Trotzdem ist die Liste nicht ganz nutzlos; sie eignet sich nämlich bestens als Nachschlagewerk. Sollten Sie zum Beispiel einmal vergessen haben, wie man die Filmsequenzmethode `gotoAndPlay()` genau schreibt (beachten Sie, dass ActionScript zwischen Groß- und Kleinschreibung unterscheidet), brauchen Sie nur den Knoten  `Globale Funktionen` und den untergeordneten Knoten `Zeitleistensteuerung` zu erweitern und die Liste der Methoden zu durchsuchen. Wenn Sie wissen wollen, welche Eigenschaften es für Filmsequenzen gibt, erweitern Sie einfach den Knoten `Integrierte Klassen\Film\MovieClip\Eigenschaften`.

Wenn Sie die Maus für einen Moment über einem Element der Aktionen-Liste verweilen lassen, springt sogar ein kleines Fenster mit einer Kurzbeschreibung des Elements auf.



Ob das Fenster aber eine wirkliche Arbeitserleichterung darstellt, scheint mir zumindest zweifelhaft. Ich persönlich ziehe es vor, das Fenster auszublenden (Klick auf den Pfeilschalter im unteren Rahmenelement) und so mehr Platz für den Skript-Navigator zu schaffen. Als Schnellreferenz nutze ich die erste Schaltfläche aus der Symbolleiste des Skriptfensters (`Skriptobjekt hinzufügen`), die – bis auf die Kurzbeschreibungen – die gleiche Funktionalität bietet.

### 3.1.2 Code zuordnen

Das AKTIONEN-Bedienfeld verbindet den Code, den Sie in das rechte Teilfenster eingeben, stets mit dem Element, das auf der Bühne ausgewählt ist. Umgekehrt wird, wenn Sie ein Element auf der Bühne auswählen, im AKTIONEN-Bedienfeld der Code angezeigt, der mit dem Element verbunden ist.

Wenn Sie also Code für ein Element aufsetzen oder den Code eines Elements überarbeiten wollen, gehen Sie so vor, dass Sie zuerst das Element auswählen und dann in das AKTIONEN-Bedienfeld wechseln. Wie dies für Schlüsselbilder, Schaltfläche-, Komponenten- und Filmsequenzinstanzen im Detail aussieht, haben Sie bereits in Kapitel 2 erfahren.

Wenn Sie bereits bestehenden Code bearbeiten wollen, können Sie die zugehörigen Elemente direkt über den Skript-Navigator auswählen und müssen nicht erst zur Bühne wechseln. Code in Symbolen kann allerdings nicht auf diesem Weg geladen werden.

Nichts ist verwirrender, als Ereignisbehandlungscode versehentlich mit dem falschen Element zu verbinden. Kontrollieren Sie daher stets vor dem Aufsetzen/Bearbeiten des Codes, ob das Skriptfenster dem korrekten Element zugeordnet ist. Das AKTIONEN-Bedienfeld hilft Ihnen dabei, indem es das ausgewählte Element im Register am unteren Rand des Skriptfensters sowie im Skript-Navigator unter der Rubrik AKTUELLE AUSWAHL anzeigt.



### 3.1.3 Das Skripte-Register

Wenn Sie häufiger zwischen verschiedenen Codeblöcken hin und her wechseln müssen, empfiehlt es sich, die Codeblöcke permanent zu laden und über das Skripte-Register anzusteuern.

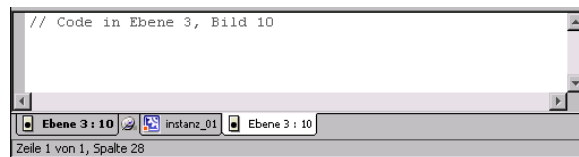


Abb. 3.3:  
Die Reiter des Skripte-Registers am unteren Rand des Skriptfensters

Das Skripte-Register ist zweigeteilt. Links steht ein einzelner Reiter, der zu dem aktuell geladenen Code gehört. Rechts davon, hinter der Pinnnadel, werden die Reiter der permanent geladenen Codeblöcke angeordnet. Den aktiven Reiter, dessen Code im Skriptfenster zu sehen ist, erkennen Sie am weißen Hintergrund.



ActionScript-Code kann festgepinnt werden.

Die Pinnnadel versinnbildlicht und steuert den Pinnzustand des aktuellen Reiters. Wenn Sie per einfachem Mausklick ein Schlüsselbild oder eine Instanz im Skript-Navigator oder in der Zeitleiste auswählen, wird der zugehörige Code – sofern vorhanden – in das Skriptfenster geladen und dort mit dem Reiter links der Pinnnadel verbunden. Um für diesen aktuell angezeigten Code einen permanenten Reiter einzurichten, müssen Sie nur auf das Symbol der Pinnnadel klicken, deren Darstellung sich daraufhin dreht, bis nur noch der Nadelkopf in der Aufsicht zu sehen ist (alternativ können Sie im Bedienfeld-Menü den Befehl **SKRIPT IMMER VORNE** aufrufen).

- Wenn Sie den Code eines Elements direkt permanent laden möchten, brauchen Sie nur im Skript-Navigator auf das Element doppelt zu klicken.
- Um einen permanenten Reiter aufzulösen, wählen Sie ihn aus und klicken Sie danach auf die Pinnnadel oder rufen Sie im Kontextmenü des Skripte-Registers den Befehl **SKRIPT SCHLIEßEN** auf.
- Um alle permanenten Reiter auf einmal zu entfernen, rufen Sie im Kontextmenü des Skripte-Registers den Befehl **ALLE SKRIPTS SCHLIEßEN** auf.



**Achtung!** Die Pinnnadel bezieht sich stets auf den aktiven Reiter (erkennbar am weißen Hintergrund), und nicht etwa auf den Reiter links neben ihr.

## 3.2 Externer Code

Es sei nicht verschwiegen, dass es auch möglich ist, ActionScript-Code außerhalb von Filmen in eigenen ActionScript-Dateien (Extension `.as`) zu speichern.

Solche externen Skriptdateien können Sie anlegen, indem Sie

- im **AKTIONEN**-Bedienfeld den Befehl **SKRIPT EXPORTIEREN** des Titelleistenmenüs aufrufen (und damit den aktuellen Inhalt des Skriptfensters in einer externen ActionScript-Datei speichern) oder
- im Dialogfenster **NEUES DOKUMENT** (Aufruf über den Menübefehl **DATEI/NEU**) die Option **ACTIONSCRIPT-DOKUMENT** wählen (und damit eine neue Skriptdatei anlegen und zur Bearbeitung in den Editor für externe Skriptdateien laden)

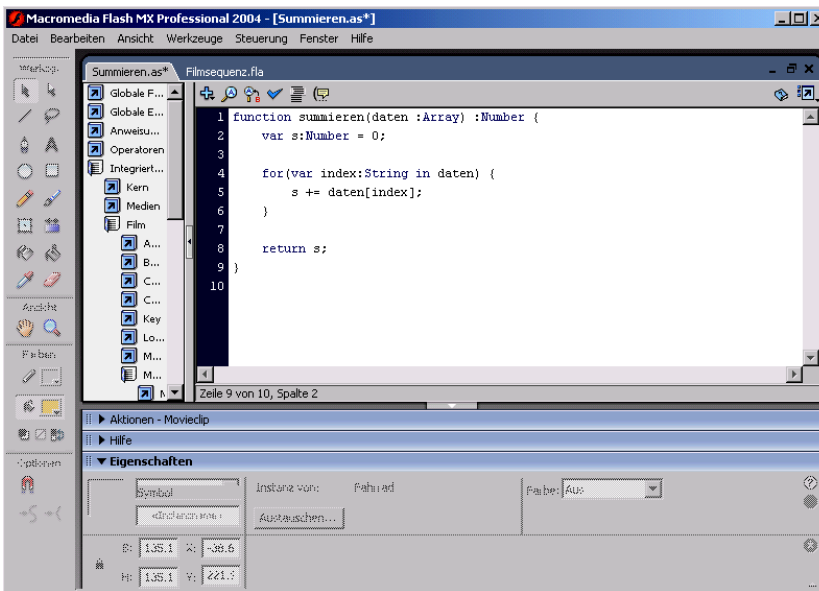


Abb. 3.4: Bearbeitung einer externen Skriptdatei im Skriptfenster. Die filmspezifischen Elemente der Authoring-Umgebung sind deaktiviert.

Allerdings eignet sich nicht jeder x-beliebige ActionScript-Code für die Auslagerung in eine externe Datei. Sinnvoll ist vor allem die Auslagerung von allgemein nützlichem Code, der vorzugsweise in eine Funktion (siehe Kapitel 8) oder eine Klasse (siehe Kapitel 9) verpackt sein sollte. Im Kapitel 8 werden Sie hierzu noch ein kleines Beispiel sehen.

### 3.3 Das Ausgabefenster

Zur Flash-Programmierungsumgebung gehört auch das Ausgabefenster. Der Flash Player nutzt dieses Fenster, um Sie auf Fehler in Ihrem ActionScript-Code hinzuweisen.

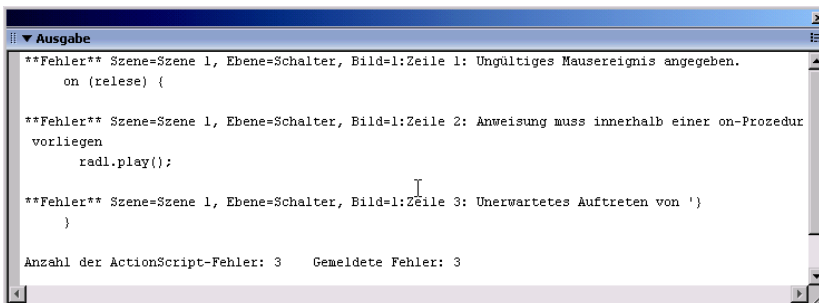


Abb. 3.5: Fehlermeldungen im Ausgabefenster

Wenn Sie beim Testen eines Flash-Films mit einer solchen Fehlermeldung konfrontiert werden, gehen Sie wie folgt vor:

1. Sehen Sie sich die erste Fehlermeldung an.

Lesen Sie sich die Beschreibung durch und schauen Sie sich den zugehörigen Code an. In dem Code aus Abbildung 3.5 fehlt zum Beispiel das »a« in »release«.



Wenn Sie in der betreffenden Zeile partout keinen Fehler entdecken können, wechseln Sie zu Flash und sehen Sie sich im AKTIONEN-Bedienfeld den Code vor der bemängelten Zeile an.

2. Korrigieren Sie den Fehler im ActionScript-Code.
3. Lassen Sie den Film erneut ausführen. Gibt es weitere Fehlermeldungen, fangen Sie wieder bei Schritt 1 an, falls nicht ...umso besser.

Vielleicht fragen Sie sich, warum Sie nicht gleich alle Fehler korrigieren sollen? Nur allzu oft beziehen sich die weiteren Fehlermeldungen auf Folgefehler, die nicht mehr auftreten, wenn der erste Fehler korrigiert wurde. In Abbildung 3.5 treten die zweite und dritte Fehlermeldung zum Beispiel nur deswegen auf, weil die erste Zeile keine korrekte Ereignisbehandlungsdefinition einleitete. Sollten die nachfolgenden Fehlermeldungen auf echte, weitere Fehler hinweisen, werden Sie beim nächsten Testlauf wieder auftreten.



Nach einer gewissen Zeit werden Sie einen Blick dafür bekommen, ob nachfolgende Fehlermeldungen auf Folgefehler oder eigenständige Fehler hinweisen.

### Eigene Ausgaben

Mithilfe der Funktion `trace()` können Sie selbst Daten in das Ausgabefenster ausgeben. Sie brauchen der Funktion nur den auszugebenden Text (in doppelten Anführungszeichen) oder Wert in runden Klammern zu übergeben:

```
trace("Hallo");
trace(radinstanz._x);
```

Mit dem Plus-Operator können Sie die Ausgabe von Texten und Werten kombinieren:

```
trace("Wert von radinstanz._x: " + radinstanz._x);
```

Doch warum sollten Sie dies tun? Bei Ausführung des Flash-Films im Browser eines Webbesuchers wird kein Ausgabefenster angezeigt, ja die `trace()`-Auf-

rufe werden vom Flash Player sogar vollständig ignoriert – lediglich bei Testläufen in Flash werden sie verarbeitet.

Nun, auch wenn `trace()` keinen Einfluss auf den eigentlichen Flash-Film hat, ist es doch ein wunderbares Hilfsmittel, um während der Entwicklungsphase des Films zu kontrollieren, was der erstellte ActionScript-Code eigentlich macht und wie er ausgeführt wird.

Betrachten Sie noch einmal das Fahrrad-Beispiel aus Kapitel 2.5.3. In dem Beispiel wurde das Fahrrad, bei dem es sich um die Instanz eines Filmsequenzsymbols handelte, entlang eines Pfads animiert. Nehmen wir an, Sie sollen zu diesem Film Code schreiben, der das Fahrrad so lange anhält, wie die Maustaste gedrückt wird. In einem ersten Versuch könnten Sie folgenden Code mit der Fahrrad-Instanz verbinden:

```
onClipEvent (mouseDown) {  
    stop();  
}
```

```
onClipEvent (mouseUp) {  
    play();  
}
```

Wenn Sie den Film danach ausführen und testen, können Sie so oft in den Film klicken, wie Sie wollen – es geschieht nichts. Woran kann dies liegen? Zum einen könnte es sein, dass das Ereignis gar nicht abgefangen und der Code bei Eintritt des Ereignisses nicht ausgeführt wird. Genauso gut kann es aber sein, dass der Code, d.h. die Funktionsaufrufe `stop()` und `play()`, falsch sind. Mithilfe von `trace()`-Aufrufen können Sie sich schnell Klarheit verschaffen, Sie brauchen nur zwei passende Aufrufe in den Ereignisbehandlungscode einzufügen:

```
onClipEvent (mouseDown) {  
    stop();  
    trace("gedrückt");  
}
```

```
onClipEvent (mouseUp) {  
    play();  
    trace("losgelassen");  
}
```

Wenn Sie den Film jetzt erneut testen, wird das Fahrrad zwar immer noch nicht angehalten, doch bei jedem Drücken und Loslassen der Maustaste erscheinen die entsprechenden Meldungen im Ausgabefenster. Die Verbindung von Ereignis und Behandlungscode funktioniert also, der Fehler muss mit den Funktionsaufrufen `stop()` und `play()` zu tun haben. Ja, richtig! Die beiden Funktionen beziehen sich ja auf die aktuelle Filmsequenzinstanz (das Fahrrad).

Zum Anhalten des Fahrrades muss aber die Hauptfilmsequenz mit der Pfadanimation angehalten werden:

```
onClipEvent (mouseDown) {
    _parent.stop();
}

onClipEvent (mouseUp) {
    _parent.play();
}
```



Mit `trace()` können Sie nicht nur kontrollieren, ob und wann welcher Ereignisbehandlungscode ausgeführt wird. Sie können auch die Werte von (Filmsequenzinstanz-)Eigenschaften oder Variablen (siehe Kapitel 4) ausgeben und überwachen.

## 3.4 Testen und Debuggen

Spöttische Zeitgenossen behaupten, dass die eigentliche Arbeit des Programmierers erst dann beginnt, wenn der Code fertig aufgesetzt ist, und spielen damit auf die leider nicht zu leugnende Tatsache an, dass kein Programmierer unfehlbar ist und kein Programmcode geschrieben wird, der nicht den einen oder anderen Fehler enthielte. Zur Pflicht des Programmierers gehört es daher, seinen Code auch gewissenhaft zu testen:

- auf korrekte Syntax
- auf Ausführbarkeit
- auf logische Fehler

### 3.4.1 Syntaxprüfung

Einzelne Skripte, d.h. Code, der mit einer Instanz oder einem Schlüsselbild verbunden ist, können (und sollten) Sie direkt nach Fertigstellung, noch im AKTIONEN-Bedienfeld, prüfen.



Drücken Sie dazu einfach in der Symbolleiste des AKTIONEN-Bedienfelds auf die Schaltfläche SYNTAX ÜBERPRÜFEN.

Ein Dialogfenster zeigt Ihnen danach das Ergebnis der Syntaxprüfung an; etwaige Fehlermeldungen werden ins Ausgabefenster geschrieben.

### 3.4.2 Filme und Skriptcode testen

Um zu testen, ob der von Ihnen bearbeitete Flash-Film wie gewünscht abgespielt wird, rufen Sie den Befehl `STEUERUNG/FILM TESTEN` auf oder Sie drücken die zugehörige Tastenkombination `[Strg] + [↵]`. (Bei der Bearbeitung größerer Filme mit mehreren Szenen, werden Sie die Szenen wahrscheinlich einzeln mithilfe des Befehls `STEUERUNG/SZENE TESTEN` überprüfen.)

Film testen

`[Strg] + [↵]`

Sie können einen Film auch durch Aufruf des Befehls `STEUERUNG/ABSPIELEN` (`[↵]`) testen, doch wird dann nur die aktuelle Filmsequenz abgespielt. Eingebettete Filmsequenzinstanzen werden nur als statische Bilder dargestellt.



Auf die gleiche Weise testen Sie, ob der von Ihnen aufgesetzte ActionScript-Code richtig verarbeitet wird. Wenn Sie Pech haben, erscheint neben dem Film sogleich das Ausgabefenster mit einer oder mehreren Fehlermeldungen (siehe Abschnitt 3.3). Aber auch wenn keine Fehlermeldungen erscheinen, bedeutet dies nicht, dass Ihr ActionScript-Code fehlerfrei ist. Vor allem zwei Kategorien von Fehlern sind es, die – vom Flash Player schweigend akzeptiert – dem Programmierer das Leben schwer machen:

- Tippfehler in den Namen von Instanzen, Eigenschaften, Variablen, Funktionen und Methoden.

Schnell hat man statt `play()` die Zeichenfolge `p1ey()` eingetippt. Das Tragische ist, dass der Flash Player auf solche Fehler nicht hinweist. Er erkennt zwar, dass hier eine Funktion aufgerufen werden soll, die es gar nicht gibt, doch führt dies nur dazu, dass er den Aufruf ignoriert.

- Logische Fehler. Von logischen Fehlern spricht man, wenn Code syntaktisch korrekt ist, aber leider nicht das tut, was der Programmierer beabsichtigte.

Stellen Sie sich vor, Sie haben einen Flash-Film zwei Filmsequenzinstanzen `rad1` und `rad2` und zwei Schaltflächen zum Anhalten der Filmsequenzinstanzen vorliegen. Das `release`-Ereignis der ersten Schaltfläche verbinden Sie mit dem Aufruf der Methode `rad1.play()`. Dann kopieren Sie diesen Ereignisbehandlungscode und verbinden ihn mit der zweiten Schaltfläche. Sollten Sie nun vergessen, im kopierten Code den Instanznamen `rad1` in `rad2` zu ändern, haben Sie einen logischen Fehler erzeugt, denn beide Schaltflächen halten danach das erste Rad an.

Um solche Fehler aufzuspüren, müssen Sie den ActionScript-Code des Films debuggen.

### 3.4.3 Debuggen

Logische Fehler können sehr einfach oder auch sehr schwer zu beheben sein. Ob leicht oder schwer zu beheben, eines haben logische Fehler und Tippfehler gemeinsam: Sie müssen sie erst einmal finden, bevor Sie sie beheben können. Und genau dies ist meist die Hauptschwierigkeit.

Glücklicherweise stehen Ihnen bei der Fehlerlokalisierung vier leistungsstarke Hilfsmittel zur Seite:

- der Testbefehl
- die `trace()`-Funktion
- der Debugger

#### Testen, testen, testen

Das wichtigste Hilfsmittel ist der Menübefehl `STEUERUNG/FILM TESTEN`. Nutzen Sie ihn regelmäßig. Warten Sie mit dem Testen nicht, bis Sie den kompletten Code aufgesetzt haben. Wenn Sie ein Dutzend ungeprüfter Ereignisbehandlungsdefinitionen geschrieben haben und dann beim Testen ein logischer Fehler auftritt, ist es unter Umständen recht schwer, den Fehler aufzuspüren.

Besser ist es, nach jeder Ereignisbehandlungsdefinition zu testen. Tritt dabei ein Fehler auf, können Sie davon ausgehen, dass der Fehler mit dem zuletzt bearbeiteten Ereignis zusammenhängt. Ereignisse, die semantisch zusammengehören (beispielsweise das Drücken und Loslassen der Maus im Fahrrad-Beispiel aus Kapitel 2.4.3) und nicht zu umfangreich sind, können auch zusammen getestet werden.

#### Fehlersuche mit `trace()`

Mithilfe der `trace()`-Funktion können Sie kontrollieren, wann welche Codeblöcke ausgeführt werden, und sich die Werte von Variablen und Eigenschaften anzeigen lassen (siehe Abschnitt 3.3).

#### Der Debugger

Der Debugger ist das leistungsfähigste Werkzeug zur Lokalisierung von logischen Fehlern. Mit seiner Hilfe können Sie einen Film an einer beliebigen ActionScript-Anweisung anhalten, die Werte von Filmsequenzeigenschaften und selbst definierten Variablen (siehe Kapitel 4) kontrollieren, den Film schrittweise ausführen.

##### 1. Haltepunkte setzen (Breakpoints)

Um den Film bei der Ausführung im Debugger gezielt an einem bestimmten Punkt des ActionScript-Codes anzuhalten, müssen Sie in der betreffenden Zeile einen Haltepunkt setzen.

Laden Sie den Code mit der betreffenden Anweisung dazu in das AKTIONEN-Bedienfeld und klicken Sie links neben der Zeile mit der Anweisung in die Debugger-Leiste. Ein roter Punkt zeigt fortan an, dass in die Zeile ein Haltepunkt gesetzt wurde.

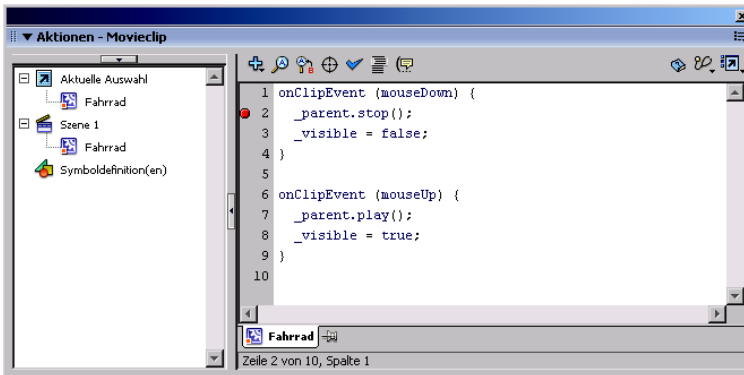


Abb. 3.6:  
ActionScript-  
Code mit  
Haltepunkt

## 2. Starten Sie den Debugger

Um den Debugger aufzurufen, rufen Sie den Menübefehl STEUERUNG/DEBUGGEN auf.

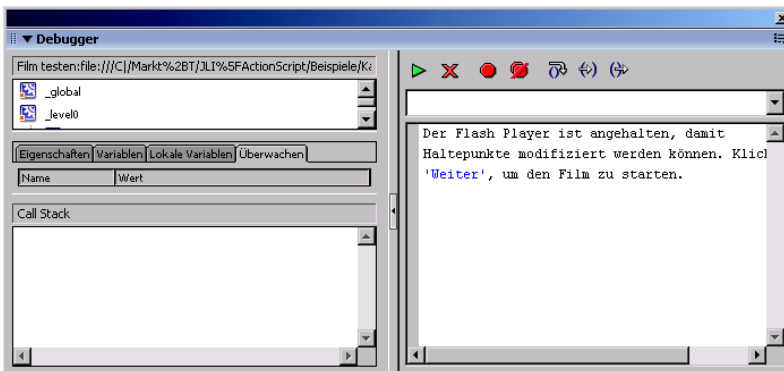


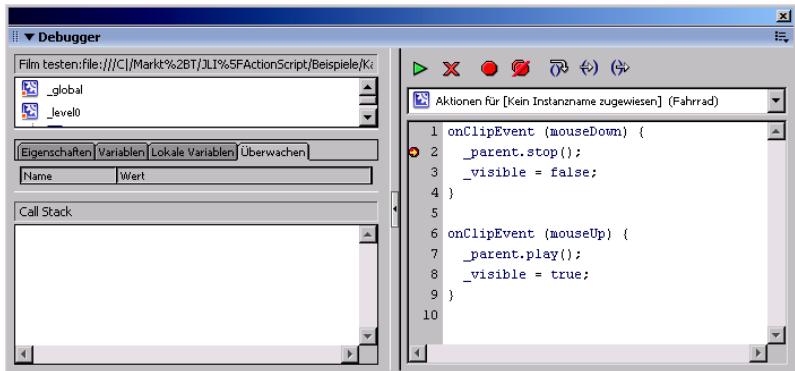
Abb. 3.7:  
Der Debugger  
wurde gestartet

Klicken Sie jetzt auf die Schaltfläche WEITER, um den Film auszuführen.



Stößt der Debugger während der Filmausführung auf einen Haltepunkt unterbricht er die Filmausführung (siehe Abbildung 3.8). Beachten Sie, dass Haltepunkte in Ereignisbehandlungsdefinitionen natürlich nur erreicht werden, wenn im Film das betreffende Ereignis ausgelöst wird!

Abb. 3.8:  
Der Film wurde  
am Halte-  
punkt ange-  
halten



Einmal angehalten, können Sie den Debugger dazu nutzen, sich die aktuellen Werte von Eigenschaften und Variablen anzuschauen.

3. Kontrollieren Sie wichtige Eigenschaften und Variablen.

Zur Überwachung von Variablen gibt es im Debugger vier Registerseiten:

Tabelle 3.1:  
Variablen  
überwachen

Registerseite	Beschreibung
EIGENSCHAFTEN	Auf dieser Seite werden die Eigenschaften einer Filmsequenzinstanz angezeigt. Wählen Sie die betreffende Instanz in dem darüber liegenden Fenster aus.
VARIABLEN	Auf dieser Seite werden die Variablen einer Filmsequenzinstanz angezeigt. Wählen Sie die betreffende Instanz in dem darüber liegenden Fenster aus.
LOKALE VARIABLEN	Auf dieser Seite werden die lokalen Variablen der aktuellen Funktion/Methode angezeigt.
ÜBERWACHEN	Auf dieser Seite können Sie Variablen aus dem aktuellen Gültigkeitsbereich anzeigen lassen. Rufen Sie dazu im Kontextmenü der Seite den Befehl HINZUFÜGEN auf, doppelklicken Sie in das Feld unter der Spalte NAME und geben Sie den Namen der Variablen ein.



Filmsequenzinstanzen, denen Sie im Eigenschafteninspektor keine Namen zugewiesen haben, werden rechts oben im Debuggerfenster unter automatisch generierten Namen aufgeführt. Die Zuordnung der Instanzen wird dadurch erschwert. Es empfiehlt sich daher, allen Instanzen vor dem Debuggen Namen zuzuweisen.

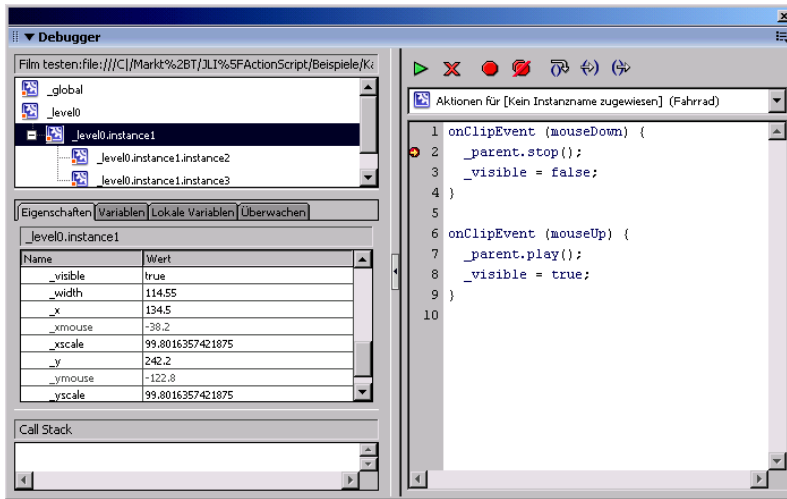


Abb. 3.9: Kontrolle der Eigenschaften von instance1 (die Fahrrad-Instanz)

4. Führen Sie den Filmcode schrittweise aus und achten Sie darauf, wie sich die Werte von Eigenschaften und Variablen ändern und ob der Code in der richtigen Abfolge ausgeführt wird.





Symbol	Befehl	Beschreibung
	WEITER	Führt den Code weiter aus, bis zum nächsten Haltepunkt oder dem Ende des Films.
	ÜBERSPRINGEN	Führt nur die nächste Anweisung aus. Handelt es sich dabei um den Aufruf einer Funktion oder Methode wird diese in einem Schritt ausgeführt.
	HINEINSPRINGEN	Führt nur die nächste Anweisung aus. Handelt es sich dabei um den Aufruf einer Funktion oder Methode, springt der Debugger in die erste Anweisung der Funktion/Methode und stoppt dort.
	VERLASSEN	Führt die aktuelle Funktion/Methode bis zum Ende aus.

Tabelle 3.2: Debugger-Befehle zur schrittweisen Ausführung

Wenn Sie den gesuchten Fehler mithilfe des Debuggers auffindig gemacht haben, beenden Sie die Debug-Sitzung und kehren zurück ins AKTIONEN-Bedienfeld, wo Sie den Fehler korrigieren.

5. Beenden Sie die Debug-Sitzung durch Klick auf die Symbolschaltfläche DEBUGGEN BEENDEN.



## 3.5 Compiler und Interpreter

Bisher war immer davon die Rede, dass der Flash Player den ActionScript-Code ausführt. Dies ist jedoch nicht ganz richtig.

### Der Interpreter

In den Flash Player ist ein besonderes Programm integriert, das Interpreter genannt wird. Dieser Interpreter ist es, der den ActionScript-Code liest und Anweisung für Anweisung ausführt.

Wenn ein Websurfer eine Webseite mit einem Flash-Film ansteuert, geschieht demnach Folgendes:

1. Die Webseite wird in den Browser des Websurfers geladen.
2. Der in der Webseite enthaltene Flash-Film wird in den Flash Player geladen, der (hoffentlich) in den Webbrowser integriert ist.
3. Der Flash Player beginnt mit dem Abspielen des Films.
4. Tritt ein Ereignis ein, für das Ereignisbehandlungscode vorgesehen ist, liest der Interpreter den Code ein, wandelt ihn in Flash-Befehle um und sorgt für deren Ausführung.

Wird der Abspielkopf des Flash Players auf ein Schlüsselbild mit ActionScript-Code gerückt, wird zuerst der ActionScript-Code ausgeführt und danach der Inhalt des Bildes angezeigt (es sei denn, der ActionScript-Code hat den Abspielkopf zuvor auf ein anderes Bild gesetzt).

### Der Compiler

Auch wenn der Interpreter Ihren ActionScript-Code ausführt, den zugehörigen Quelltext, den Sie in das AKTIONEN-Bedienfeld eingetippt haben, bekommt er nicht zu sehen. Wie das?

Wenn Sie den Film veröffentlichen oder testen, wird aus der FLA-Datei des Films eine SWF-Datei erzeugt. Wie Sie sicherlich wissen, ist dies die Datei, die Sie in Ihre Webseiten einbetten und die der Flash Player abspielt. In der SWF-Datei steht Ihr ActionScript-Code aber nicht mehr als Quelltext, sondern als binärer Bytecode.

Im Zuge der Erzeugung der SWF-Datei wurde nämlich ein weiteres internes Hilfsprogramm, der so genannte Compiler, aufgerufen. Dessen Aufgabe ist es, Ihre ActionScript-Skripte in Bytecode<sup>1</sup> zu übersetzen.

---

1. Bytecode ist prozessorunabhängiger Maschinencode.

Dies hat verschiedene Vorteile:

- Der Compiler prüft den Code auf syntaktische Korrektheit und macht Sie auf syntaktische Fehler aufmerksam.
- Der binäre Bytecode kann vom Interpreter schneller ausgeführt werden als dies für den ursprünglichen Quelltext der Fall wäre.
- Ihr Quelltext steht nur in kodierter, und damit geschützter Form in der SWF-Datei und kann nicht so leicht von anderen Programmierern kopiert werden.

## 3.6 Zusammenfassung

Das wichtigste Hilfsmittel des ActionScript-Programmierers ist das AKTIONEN-Bedienfeld.

Über das AKTIONEN-Bedienfeld weisen Sie Schlüsselbildern und Instanzen ActionScript-Code zu. Auch bestehenden Code bearbeiten Sie im AKTIONEN-Bedienfeld. Skript-Navigator und Skripte-Register helfen Ihnen, schnell und sicher zwischen bestehenden Skripten hin und her zu wechseln.

Nur fehlerfreier Code ist guter Code. Syntaxprüfung, `trace()`-Funktion und Debugger helfen, Fehler aufzuspüren und auszumerzen.

## 3.7 Fragen und Antworten

**F: Ich habe bereits früher, mit Flash MX, ein wenig mit ActionScript programmiert. Damals musste ich das AKTIONEN-Bedienfeld immer explizit in den Entwicklermodus umschalten. Gibt es diesen Modus noch?**

A: Nein, die Trennung in Normal- und Entwicklermodus wurde mit der 2004-Version aufgehoben.

## 3.8 Übungen

1. Auf der Buch-CD finden Sie im Unterverzeichnis *Kapitel03* die Filmdatei *Filmsequenz fla*, die einen Fehler enthält: Der Film reagiert nicht wie gewünscht auf Mausclicks (das Fahrrad sollte ein- und ausgeblendet werden). Der Fehler ist nicht schwer zu finden, ein intensiver Blick auf den Ereignisbehandlungscode sollte eigentlich genügen. Trotzdem sollten Sie zur Übung systematisch vorgehen, und den Fehler mithilfe von `trace()`-Anweisungen oder unter Verwendung des Debuggers aufspüren. Kontrol-

lieren Sie zuerst, ob der Ereignisbehandlungscode überhaupt ausgeführt wird. Wenn ja, prüfen Sie, wie sich der Wert der Eigenschaft `_visible` ändert.