```
            var left = ( i + this.currentBlock.x == -1 );
            if ( solid && ( right || left || occpied || bottom ) )
            this.currentRotate = copy;
         }
      }
      this.screen.update ();
}
```

Ganz unten steht der Aufruf, wie oben erklärt, und dann startet Tetris von selbst!

```
Game = new Tetris ( 20 , 10 , blocks );

//
```

Highscore mit Flash MX und PHP/MySQL

Einleitung

◀ **boblgum**

Neues Spiel, neues Glück. Im letzten Buch haben wir eine mögliche Lösung vorgestellt, wie man eine Highscore-Liste in Flash realisieren kann. Im Unterschied zum »alten« Tutorial werden wir diesmal versuchen, neue Features von Flash MX zu nutzen. Des Weiteren wollen wir nun eine Datenbank statt einer Textdatei verwenden.

Die zwei wichtigsten Neuerungen, die in diesem Tutorial verwendet wurden, sind das `LoadVars`-Objekt und das `TextField`-Objekt.

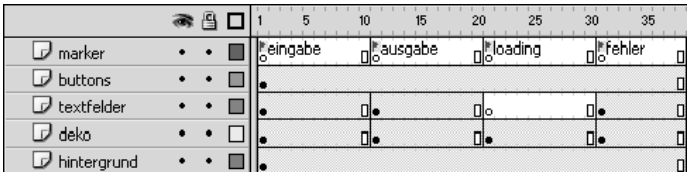
Aufbau des Films

Da Flash MX noch bessere Möglichkeiten zum objektorientierten Programmieren bietet, vereinfachen sich der Aufbau und somit auch die spätere Wartung des gesamten Films. Das komplette ActionScript (in unseren Fall sind es nur 78 Zeilen Code) befindet sich im ersten Schlüsselbild der Hauptzeitleiste, womit sich die Hauptzeitleiste auf ein einzelnes Schlüsselbild reduziert. Zur besseren Übersicht legen wir aber zwei Ebenen an. In die zweite Ebene kommt unser Form-MovieClip (`mcForm`) (der Instanzname).

mcForm

Damit man später nur noch das Aussehen zu verändern braucht, verwenden wir einen MovieClip. Man könnte aber auch alles direkt auf dem `_root` erstellen.

In der Zeitleiste befinden sich fünf Ebenen. Damit man die Bildbezeichnungen besser erkennen kann, haben wir einige Bilder extra angelegt. Im Grunde genommen lässt sich der Aufbau auf vier aufeinander folgende Schlüsselbilder reduzieren.

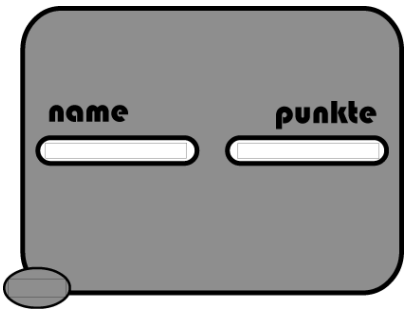


▲ **Abbildung 19**
Zeitleiste des »mcForm«-MovieClips

Aus Abbildung 19 kann man auch den logischen Aufbau der ganzen Geschichte erkennen. Die Bildbezeichnungen in der »marker«-Ebene stellen vier mögliche Zustände dar, die eintreffen könnten:

Die Abbildungen 20 – 23 erläutern die Bildbezeichnungen:

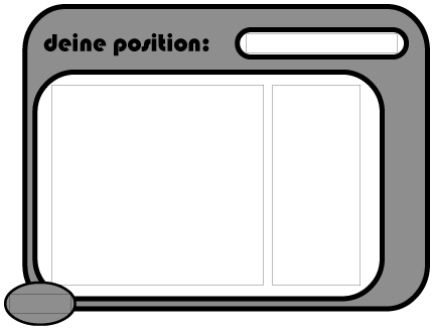
1. **Eingabe:** Hier soll der Benutzer seinen Namen (`txtName`) eingeben, und es wird die erreichte Punktezahl (`txtPunkte`) angezeigt.



▲ **Abbildung 20**
Möglicher Zustand der Marker-Ebene: Eingabe

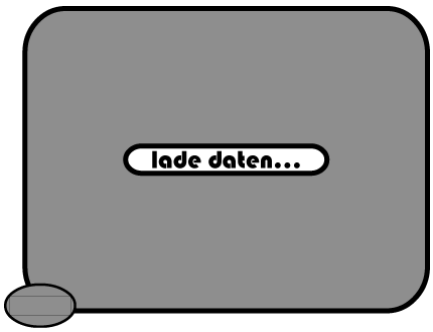
2. **Ausgabe:** Hier werden die Highscore-Liste und die Position des Benutzers (`txtDeinePosition`) angezeigt. Beachten Sie, dass die Anzeige der Liste auf zwei Textfelder verteilt wird. Dabei werden im linken Textfeld

(`txtRanklist`) die Positionen und die Namen und im rechten Textfeld (`txtPointlist`) die Punkte angezeigt. Die beiden Textfelder sollten HTML-fähig sein. Dazu aktiviert man die Option TEXT ALS HTML WIEDERGEBEN. Hier wird auch die vorgefertigte ScrollBalken-Komponente verwendet, die das `txtRanklist` steuert. Aus technischen Gründen ist sie nicht auf dem Bild zu sehen.



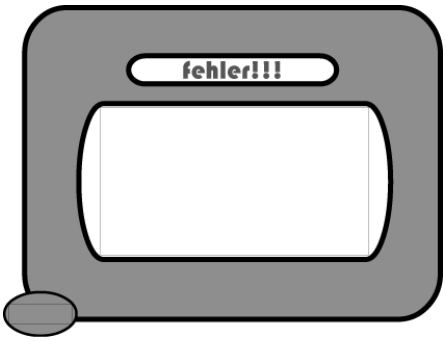
▲ **Abbildung 21**
Möglicher Zustand der Marker-Ebene: Ausgabe

3. **Loading:** Dieser Screen wird angezeigt, während unser Film die Daten an den Server übermittelt und auf die Rückgabe wartet.



▲ **Abbildung 22**
Möglicher Zustand der Marker-Ebene: Loading

4. **Fehler:** Falls »unterwegs« irgendein Fehler auftreten sollte, wird der Film zu diesem Screen umgeleitet. Die Fehlermeldungen (`txtFehler`) werden vom PHP-Script generiert und an Flash übergeben.



▲ **Abbildung 23**
Möglicher Zustand der Marker-Ebene: Fehler

Zu beachten wäre noch, dass man die Eigenschaft der Textfelder, in die der Benutzer schreiben darf, auf EINGABETEXT und die Eigenschaft der anderen auf DYNAMISCHER TEXT setzt.

mcButton

Abgesehen von den Textfeldern befindet sich ein weiterer wichtiger Bestandteil des Films im MovieClip `mcForm`. Das ist die »Steuereinheit« des ganzen Films – `mcButton`. Über die gesamte Länge des MovieClips `mcForm` benötigen wir nur einen einzigen Button (siehe Abbildung 19, Ebene Button). Der Aufbau des Buttons, der eigentlich gar keiner ist, ist simpel.

▲ **Abbildung 24**
Zeitleiste des »mcButton«-MovieClips

Die erste Ebene ist für die Bildbezeichnungen vorgesehen. In der zweiten Ebene befindet sich ein Textfeld (`txtTopic`), in dem die Aufschrift des Buttons angezeigt wird. Die dritte Ebene ist für den grafischen Schnickschnack da. Wir haben nur zwei verschiedene Zustände verwendet. Man könnte aber auch komplette Animationen hineinpacken.

ActionScript

Schauen wir uns nun den interessanteren Teil des Tutorials an. Wie schon am Anfang angekündigt, befindet sich alles im ersten Schlüsselbild der Hauptzeitleiste. Insgesamt sind es 78 Zeilen Code, von denen die meisten aber für die Grafik zuständig sind. Bevor wir den Code Zeile für Zeile durchgehen, möchte ich die Idee hinter dem Ablauf erläutern.

Nehmen wir an, der Benutzer hat das Spiel durchlaufen und will sich in der Highscore-Liste verewigen. Was passiert? Es wird ihm ein Formular »vorgelegt«, in das er seinen Namen eingeben kann (man kann es auch erweitern). Damit er weiß, wie gut oder wie schlecht er gewesen ist, wird sein Punktestand angezeigt. Auf dem Button unten links steht SENDEN.

Nehmen wir nun an, dass der Benutzer vorschriftsmäßig seinen Namen eingetippt und auf den Button geklickt hat. Damit es ihm nicht allzu langweilig wird, »blättern« wir zu einer Seite, auf der ihm eine Grafik angezeigt wird, die ihn bittet, sich doch etwas zu gedulden (hoffentlich ist der Server nicht zu langsam), seine Daten würden gerade an den Server übermittelt. Auf dem Button ist nur »…« zu sehen, sodass der Benutzer nicht auf die Idee kommt, mitten im Sendevorgang dazwischenzufunken. Und falls doch, sind wir schlauer (siehe unten). Nach einem (bleiben wir optimistisch) kurzen Augenblick ist der Server mit der Auswertung der Daten fertig und liefert eine fertige Highscore-Liste und die Position des Benutzers (falls er gut genug gewesen ist) in dieser zurück.

Nun kann der Benutzer die Leistungen anderer Spieler bewundern und sich ausgiebig über seinen miesen Punktestand ärgern, den er an der farbigen Hervorhebung in der Liste erkannt hat. Auf dem Button ist nun die Aufschrift ZURÜCK zu sehen. Falls doch im Ablauf irgendein Fehler passieren sollte, wird dem Benutzer die Fehlermeldung präsentiert. Auf dem Button ist auch in diesem Fall ZURÜCK zu sehen, sodass der Benutzer zurück zum Eingabeformular blättern könnte.

Nun ist hoffentlich auch erkennbar, was wir alles an ActionScript brauchen. Es ist aber wirklich nicht viel.

Der PHP-Loader

Wie der Name schon sagt, ist der PHP-Loader dazu da, die Daten zwischen Flash und dem Server hin- und herzuschieben. Mit Flash MX haben wir die Möglichkeit, ein Objekt anzulegen, das genau diese Aufgabe erledigen kann: `LoadVars`. Mit

```
phpLoader = new LoadVars();
```

(2. Zeile des Scripts)

erzeugen wir es auf dem `_root`. Wieso gerade da? Man kann es auch mit `mcForm.phpLoader` direkt in unserer Form erzeugen. Das ist aber nicht zwingend.

Wenden wir uns nun den Methoden und den Instanzvariablen des neuen Objekts zu.

```
03 phpLoader.punkte = Math.round(Math.random()*10000);
04 phpLoader.link = "http://localhost/test.php";
05 phpLoader.doSend = function() {
06    this.sendAndLoad(this.link, this, "POST");
07 };
08 phpLoader.onLoad = function() {
09    if (this.loaded == true) {
10       with (mcForm) {
11          mcButton.txtTopic.text = "zurück";
12          switch (this.error) {
13          case "0" :
14             gotoAndStop("ausgabe");
15             txtRanklist.onScroller = function() {
16                txtPointlist.scroll = this.scroll;
17             };
18             txtRanklist.htmlText = this.ranklist;
19             txtPointlist.htmlText = this.pointlist;
20             txtDeinePosition.text = this.deinePosition;
21             break;
22          default :
23             gotoAndStop("fehler");
24             txtFehler.text = this.error;
25          }
26       }
27    } else {
28       with (mcForm) {
29          gotoAndStop("fehler");
30          txtFehler.text = "Flash konnte die Daten nicht
            senden!";
31       }
32    }
33 }
```

Gehen wir den Code Zeile für Zeile durch:

Zeile 03: Wir legen im Objekt eine neue Variable `punkte` an. Hier weisen wir ihr einen Zufallswert zu. Es sollte sich aber schon um echte Punkte handeln.

Zeile 04: Hier wird eine Variable `link` angelegt. Der Wert ist gleich dem Link zu unserem PHP-Script.

Zeile 05 bis 07: Hier weisen wir dem Objekt eine neue Methode `doSend` zu. Eigentlich könnte man darauf verzichten und den Befehl aus der Zeile 6 direkt aufrufen; so aber kann man auch Aktionen vor und nach dem Ladevorgang ausführen (z. B. einen Timer).

In den nächsten Zeilen steht, was passieren soll, wenn der Ladevorgang abgeschlossen wurde. Dabei unterscheiden wir zwischen zwei möglichen Zuständen: Was soll passieren, wenn der Ladevorgang ohne Fehler abgeschlossen wurde, und was soll passieren, wenn Fehler aufgetreten sind?

Zeile 9–26: Keine Fehler: Wir »sprechen« den `mcForm`-MovieClip und seine Bestandteile an. So soll unser Button in jedem Fall die Überschrift ZURÜCK bekommen. Danach wird in Abhängigkeit von den Rückgabewerten unseres PHP-Scripts entschieden:

Zeile 13–21: Falls alles okay gewesen ist, soll `mcForm` zum Bild `ausgabe` springen. Erst jetzt können wir auch die Textfelder, die sich in diesem Bild befinden, ansprechen. Beachten Sie, dass wir hier die `htmlText`-Eigenschaft verwenden, um die Rich-Text-Formatierung zu erhalten, die vom PHP-Script generiert wird.

Weiterhin wollen wir, dass, wenn im Textfeld `txtRanklist` gescrollt wird, gleichzeitig auch `txtPointlist` mitscrollt. Dazu weisen wir dem Ereignis `onScroll` des `txtRanklist` eine Funktion zu. Dann wollen wir, dass in allen Textfeldern überhaupt was angezeigt wird. Die Eigenschaft `text` der jeweiligen Textfelder hilft uns da weiter. Mit `break` verlassen wir unsere Abfrage.

Zeile 22–24: Falls vom PHP-Script doch irgendein Fehler gemeldet wurde, soll `mcForm` zum Bild `fehler` springen und im Textfeld `txtFehler` die Meldung anzeigen.

Zeile 27–32: Falls während des Ladevorgangs Fehler aufgetreten sind:

Hier kann es sich nur um systembedingte Fehler handeln (z. B. dass der Server offline ist). Der `mcForm`-MovieClip soll zum Bild `fehler` springen und eine von uns ausgedachte Fehlermeldung in `txtFehler` anzeigen.

Das sieht doch zunächst alles viel wilder aus, als es eigentlich ist, oder?

mcForm-ActionScript

Wenden wir uns dem Rest des Codes zu. Hier handelt es sich um alles, was mit `mcForm` zu tun hat.

```
35 with (mcForm) {
36    gotoAndStop("eingabe");
37    txtPunkte.text = _root.phpLoader.punkte;
38    // **********BUTTON*************
39    mcButton.gotoAndStop("out");
40    mcButton.txtTopic.text = "senden";
41    //button-actions
42    mcButton.onPress = function() {
43       this.gotoAndStop("over");
44    };
45    mcButton.onRollOver = function() {
46       this.gotoAndStop("over");
47    };
48    mcButton.onRollOut = function() {
49       this.gotoAndStop("out");
50    };
51    mcButton.onRelease = function() {
52       switch (this.txtTopic.text) {
53       case "senden" :
54          _root.phpLoader.name = this._parent.txtName.text;
55          _root.phpLoader.doSend();
56          with (this) {
57             txtTopic.text = "...";
58             _parent.gotoAndStop("loading");
59          }
60          break;
61       case "zurück" :
62          this.txtTopic.text = "senden";
63          with (this._parent) {
64             gotoAndStop("eingabe");
65             txtPunkte.text = _root.phpLoader.punkte;
66             txtName.text = _root.phpLoader.name;
67          }
```

```
68          break;
69       default :
70       }
71    };
72    mcButton.onReleaseOutside = function() {
73       with (this) {
74          gotoAndStop("out");
75          _parent.gotoAndStop(_parent._currentframe);
76       }
77    };
78}
```

Was geschieht hier? Nicht viel.

Zeile 35: wir wenden uns dem `mcForm`-MovieClip zu …

Zeile 36: Bei der Initialisierung des Films soll `mcForm` zum Bild `eingabe` springen und dort stehen bleiben.

Zeile 37: Im Textfeld `txtPunkte` wollen wir die Variable `punkte` aus unserem `phpLoader`-Objekt anzeigen lassen.

Zeile 38: Nun geht es mit den Aktionen und Eigenschaften unseres Buttons los.

Zeile 39: Am Anfang soll `mcButton` zum Bild `out` springen und dort stehen bleiben …

Zeile 40: … und wir setzen seine Aufschrift auf `senden`.

Zeile 42–77: Die Methoden des `mcButton`

Zeile 42–44: Beim Klicken und Festhalten der linken Maustaste auf `mcButton` soll dieser zum Bild `over` springen und dort stehen bleiben.

Zeile 45–47: Dasselbe soll passieren, wenn man einfach mit dem Cursor über den Button fährt.

Zeile 48–50: Sobald der Cursor die Fläche des `mcButton` verlässt, soll `mcButton` zurück zum Bild `out` springen etc.

Zeile 51–71: Hier ist der interessantere Teil des `mcButton`. Was passiert, wenn man die Maustaste (noch über dem Button!) loslässt? Hier unterscheiden wir folgende Zustände des Buttons:

Zeile 53–60: Falls die Aufschrift des Buttons SENDEN lautet, legen wir als Erstes eine Variable `name` an und setzen ihren Wert gleich dem Text im Textfeld `txtName`. Der `mcForm`-MovieClip befindet sich wohlgemerkt im Bild `eingabe`. Dann rufen wir die Methode `doSend` des `phpLoader`-Objekts

auf. An dieser Stelle hätte man auch direkt die `sendAndLoad`-Methode des `LoadVars`-Objekts aufrufen können (s.o.). Danach setzen wir die Aufschrift des Buttons auf »neutral« und lassen `mcForm` zum Bild `loading` springen. Die Sprunganweisung könnte auch in der Methode `doSend` vom `phpLoader` stehen. Dann müsste man aber auf vollständige Pfadangaben achten.

Zeile 61–68: Falls die Aufschrift des Buttons ZURÜCK lautet, ändern wir diese in SENDEN und lassen `mcForm` zum Bild `eingabe` springen. Dort (und erst dort! – also nach dem Sprung) zeigen wir in den Textfeldern die »alten« Werte der Variablen `name` und `punkte` aus dem `phpLoader`-Objekt an.

Wie man hier sieht, werden bei einer »…«-Aufschrift keine Aktionen ausgeführt. Der Benutzer kann sich also während des Ladevorgangs die Fingergelenke kaputt klicken, ohne dass im Ablauf irgendein Fehler auftritt.

Zeile 72–77: Falls der Benutzer es sich doch anders überlegt haben sollte und die Maustaste außerhalb der `mcButton`-Fläche loslässt, lassen wir unseren Button zum Bild `out` springen. Der `mcForm` bleibt dort, wo er gerade ist.

Und? Wir sind mit dem ActionScript fertig. Unglaublich, aber wahr.

PHP-Script und die MySQL-Datenbank

Da es sich hier um ein Buch über ActionScript und nicht um ein Buch über PHP/MySQL handelt, wird die Erläuterung zum PHP-Script etwas knapper ausfallen. Da aber PHP, wie auch ActionScript, eigentlich ziemlich simpel sind, ist dies wohl nicht so tragisch.

Was macht unser PHP-Script, und wie sieht unsere Datenbank aus?

Datenbank

Die Datenbank ist für unsere Testzwecke denkbar einfach aufgebaut. Sie besteht aus einer Tabelle »liste«. Diese hat drei Felder:

- »id«: als Datensatzindikator, sprich Primärindex (auto-increment). Wird bei jedem neuen Datensatz automatisch angelegt. Der Wert erhöht sich automatisch. Datentyp `Integer`.
- »name«: Hier werden die Namen der Benutzer gespeichert (`phpLoader.name`). Datentyp `Varchar`. Achten Sie darauf, dass die Zeichenanzahl groß genug ist, sonst werden die Einträge abgeschnitten.
- »punkte«: Hier werden die Punktestände gespeichert (`phpLoader.punkte`). Datentyp `Integer`.

PHP-Script

Um das Ganze nicht unnötig in die Länge zu ziehen, wird hier das Script nur als Abfolge von logischen Schritten erläutert. Zu beachten ist nur die Übergabe der Variablen an Flash, die wir durch

```
echo '&variablenname=variablenwert';
```

oder

```
echo '&flashvariable='.$phpvariable;
```

lösen. An Flash wird also eine Zeichenkette übergeben.

Der Script-Ablauf ist der Folgende:

Wir setzen die Flash-Variable `error` auf 0, weil ja noch kein Fehler aufgetreten ist. Dieser Wert soll auch bleiben, falls alles reibungslos abläuft.

Da wir PHP in der aktuellsten Version 4.2.1 verwenden, »transferieren« wir die aus Flash an das Script übergebenen Variablen in eine »normale« Form. Das sollte man zwar aus Sicherheitsgründen nicht tun, da es sich aber nur um einen Test handelt, machen wir es der Einfachheit halber.

Unser Script soll nur dann abgearbeitet werden, wenn wir von Flash auch eine Variable empfangen haben. Sonst brechen wir ab mit:

```
die('&error=Keine Werte empfangen!&');
```

Falls Variablen übermittelt wurden, versuchen wir, eine Verbindung zur Datenbank herzustellen.

Falls es klappt, gehen wir weiter. Falls nicht, brechen wir die Bearbeitung des Scripts ab und übergeben gleichzeitig den »Code« der Fehlermeldung an Flash:

```
die('&error=Verbindung zur DB konnte nicht hergestellt
werden!&');
```

Falls die Verbindung zur Datenbank steht, durchsuchen wir sie erst nach Datensätzen, die dem aus Flash übermittelten Datensatz entsprechen. Der `name` und die `punkte` müssen also gleich sein.

Funktioniert die Suchanfrage überhaupt, gehen wir weiter, sonst:

```
die('&error=Konnte nicht die DB lesen&');
```

Wurde kein identischer Datensatz gefunden, versuchen wir, einen neuen in die Datenbank zu schreiben.

Hat der Schreibzugriff funktioniert, machen wir weiter, sonst:

```
die('&error=Konnte nicht in die DB schreiben!&');
```

Nun bereiten wir die Anzeige der Highscore-Liste für Flash vor. Dazu fragen wir die ganze Tabelle ab, lassen das Ergebnis erst nach Punkten und dann nach Namen sortieren und nehmen die ersten 30 Datensätze.

Wurde die Abfrage fehlerfrei ausgeführt, machen wir uns an die Anzeige, sonst:

```
die('&error=Konnte die DB nicht lesen&');
```

Wir durchlaufen das Abfrageergebnis. Dabei beachten wir Folgendes:

- Sollten wir einen Datensatz finden, der mit den Benutzerdaten übereinstimmt, heben wir dieses Stück der gesamten Zeichenkette farblich hervor.
- Die ersten zehn Datensätze (TOP 10) werden in fetter Schrift hervorgehoben.
- Die Punkteliste und die Namensliste werden in zwei verschiedenen Variablen gespeichert.

Zur guter Letzt erfolgt die endgültige Übergabe der Variablen an Flash. Fertig!

mfg boblgum
boblgum@mysterion.de

Für Ihre Notizen