

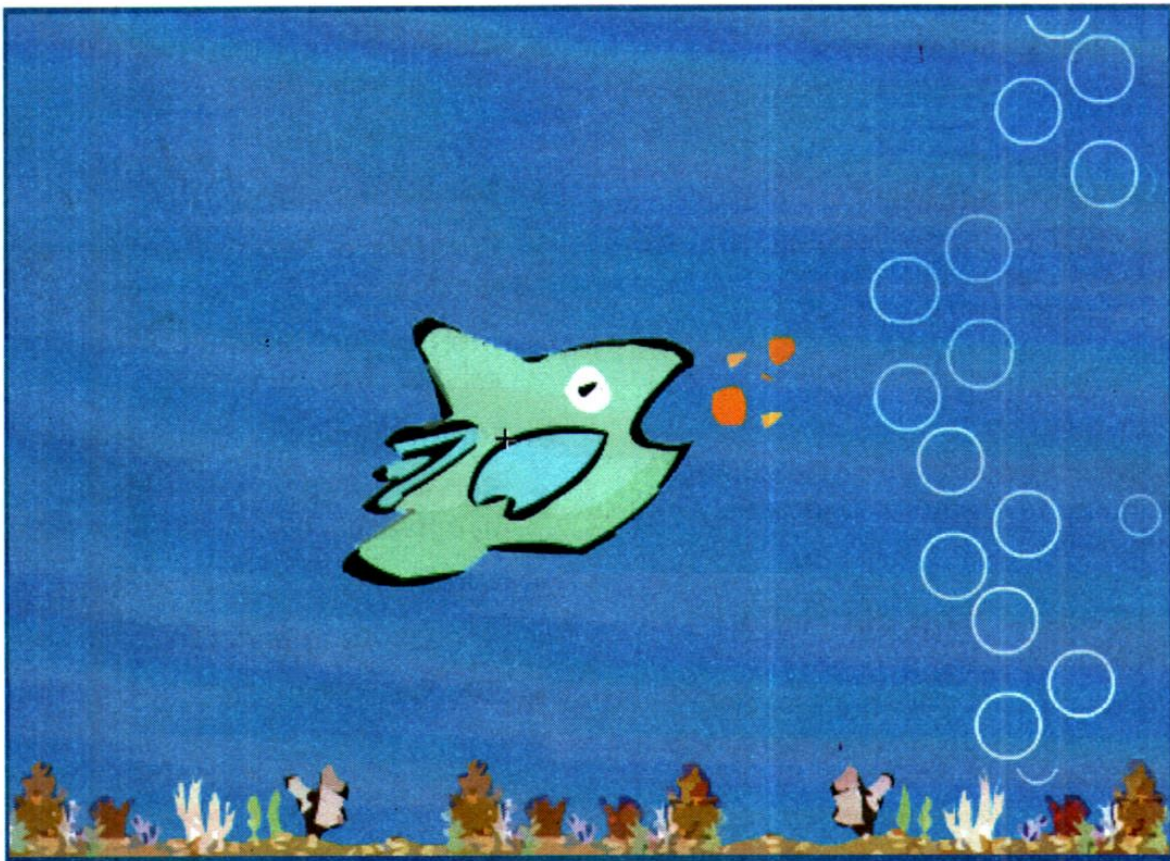
4,-

Deutschland

Ein Spiel wird programmiert ...

Spiele in Flash

ab Flash 5 - für Einsteiger



Gefressen: 24
Verloren: 2

Level 3

Fishbud

Steuerung mit der Tastatur bzw. Maus, Klonen von Objekten, Kollisionskontrolle, Punktestand, Highscoreliste, Variablenübergabe, Animationen, Sound, Preloader

www.KnowWare.de

Oliver Lange

Deutschland: 4,- EUR Österreich: 4,60 EUR
Schweiz: 8 SFR Luxemburg: 4,70 EUR Italien: 5,50 EUR



Spiele in Flash

Oliver Lange, oliverlange@knowware.de

ISBN 87-90785-97-5, 1. Ausgabe, 1. Auflage: 2002-03

© Copyright 2002, Autor und KnowWare

Michael Maardt, verlag@knowware.de - Karl Antz, lektorat@knowware.de

Printer: OTM Denmark, Binder: Gramo Denmark, Published by KnowWare

Bestellung für Endverbraucher und Vertrieb für den Buchhandel

Bonner Presse Vertrieb, Möserstr. 2-3

D-49074 Osnabrück

Tel.: +49 (0)541 33145-20

Fax: +49 (0)541 33145-33

knowware@bpv-online.com

www.knowware.de/bestellen

Vertrieb für den Zeitschriftenhandel:

IPV, Postfach 10 32 46, D-20022 Hamburg

Tel.: +49 (0) 40 23711-0

Fax: +49 (0)40 23711-215

www.ipv-online.de

Worum es geht

Hinter KnowWare steht der Gedanke, Wissen leichtverständlich und preisgünstig zu vermitteln.

Wo und wann sind die Hefte erhältlich?

Neue Hefte sind im allgemeinen zwei Monate im Handel, und zwar bei Kiosken, im Bahnhofsbuchhandel und im Buchhandel – bei vielen Verkaufsstellen sowie im Buchhandel auch länger. Alle beim Verlag vorrätigen Titel kannst du immer bestellen.

Bestellung

- bei deinem KnowWarehändler - Bestellformular am Ende des Heftes ausfüllen!
- beim Bonner Presse Vertrieb, siehe oben

www.knowware.de

- Beschreibungen und Bilder aller Hefte.
- Mehr als 100 kostenlose PDF-Dateien - zu jedem Heft gibt es eine kostenlose PDF-Datei von den ersten 15-20 Seiten
- Ausverkaufte Hefte: das ganze Heft als PDF ist kostenlos
- Geplante Hefte
- Online-Bestellung
- Kostenloser Newsletter. Viele Vorteile für dich.
- Interne Suchfunktion. Du findest schnell, was du suchst.
- Informationen für neue Autoren
- Serviceseiten zu den Heften, hauptsächlich von den Autoren selbst.
- Forum
- KnowWare in anderen Sprachen
- 500 Webseiten.

www.knowware.de

Vorwort	4	Button 2 – Ab zum Gewinnspiel	32
Das Spiel	5	Kurzübersicht Flash 5 –	
Das Gameplay ergibt das Drehbuch	5	Spielprogrammierung.....	35
Die notwendigen Grafiken:	5	Highscoreliste	39
Los geht's	6	Erweiterungen.....	44
Der 1. Akt	6	Warum schwimmt Fishbud aus dem	
Die Steuerung mit der Tastatur	8	Aquarium?.....	44
Grafikwechsel durch Tastaturklick	10	Level mit unterschiedlichen	
Variablen = Merktzettel für wichtige		Schwierigkeitsgraden	46
Informationen.....	12	Levelanzeige.....	49
Exkurs: Steuerung in zwei		Realistischer Futterfall.....	50
Richtungen	13	Bewegung im Aquarium.....	52
Exkurs: Steuerung mit der Maus	14	Sound	55
Der 2. Akt	15	Luftblasen machen Lärm	55
Das Futter wird animiert.....	15	Fishbud macht beim Fressen	
Exkurs: So arrangiert man mehrere		Geräusche.....	56
Objekte innerhalb der Zeitleiste.....	17	GameOver Signal	58
Klone des Futters erstellen	18	Der Preloader	59
Kollisionskontrolle – Fishbud frißt		1.) Info über die geladenen Bilder.....	61
das Futter	21	2.) Info über die geladenen Bytes	61
Der 3. Akt	24	Spielanleitung	64
Hochzählen von Punkten	25	Spielstart	67
Hintergrundgrafik und Spieldesign	27	Frameaufruf bei mehreren	
Spielstandabfrage – Neues Level		Szenen.....	68
oder Ende des Spiels.....	29		
Button1 – Gleich noch einmal			
spielen.....	32		

Vorwort

Durch immer schnellere Internetzugänge sind die „alten“ Grundlagen für Internetseiten ins Wanken geraten. Während früher eine Internetseite nur wenige Kilobytes haben durfte, vermittelt heute das Laden von großen Dateien, z.B. Musik und Videoclips, sowie eine immer größer werdende Anzahl von aufwendig programmierten Internetseiten ein neues Internetgefühl.

Für Programmierer und Designer bedeutet dies, dass die eigenen Projekte um neue, buntere Inhalte ergänzt werden müssen.

Neben dem Einsatz von professionellen Bildbearbeitungsprogrammen bietet sich hierfür vor allem eine Software an, die neben dem grafischen Aspekt auch zusätzliche Features wie Animationsfähigkeit, Programmierfähigkeit usw. bietet.

Das wohl bekannteste Programm in diesem Genre ist das Programm **Flash** aus dem Hause Macromedia. Hiermit können Webdesigner auf einfache Art und Weise dynamische und vor allem grafisch sehr ansprechende Internet-erweiterungen für ihre Websites und auch komplette Internetpräsentationen entwerfen.

Neben Filmen mit eindrucksvollen Effekten, eingebundenen Musik- und Soundeffekten oder animierten Menüleisten kann man Flash auch für die Programmierung von Spielen einsetzen. Und dies ist das Thema dieses Heftes!

Die einzige Voraussetzung für den Einsatz von Flash-Inhalten auf deiner Internetseite ist das Flash-Plug-In im Browser deiner Besucher. In den neueren Browserversionen ist dieser Player schon fest integriert; die Nutzer älterer Browser müssen sich den Flash-Player herunterladen.

Für das vorliegende Heft solltest du Macromedia Flash ab Version 5 haben. Wenn du diese Version nicht hast, kannst du dir auf

www.macromedia.com/de/ eine 30-Tage-Testversion herunterladen. Grundkenntnisse in Flash sind nicht unbedingt notwendig. Willst du mehr über Flash-Grundlagen wissen, empfehle ich dir das Heft Flash 5 für Einsteiger.

Es gibt im Internet eine große Anzahl von Flash-Foren und Tutorials. Als ich bei der Programmierung meines ersten Flash-Spiels auf der Suche nach einer sinnvollen Anleitung für die Spielprogrammierung war, habe ich leider feststellen müssen, dass es zwar für einzelne Themen gute Anleitungen gibt – jedoch gibt es keine komplette Hilfestellung, die z.B. Themen wie Steuerung, Kollisionskontrolle, Highscoreliste oder die Versendung einer Teilnehmer-Mail nach dem Spiel im Zusammenhang erklärt.

Dieses Heft soll diese Lücke schließen. Wir werden gemeinsam das Flash-Spiel „Fishbud“ programmieren. Dabei werde ich neben diesem Hauptziel in kleinen Exkursen auch andere Ideen und Möglichkeiten darlegen, wie z.B. die Steuerung mit der Maus – oder mit der Tastatur, usw. Im Endeffekt kannst du dir dann dein Spiel im Baukastenprinzip zusammenbauen.

Alle Grafiken und Scripte kannst du dir zur Hilfestellung auch auf www.spieleinflash.de herunterladen.

Zum Autor

Ich bin 30 Jahre alt und arbeite als Webmaster und Freiberufler. Vor gut 18 Jahren habe ich mit einem PC 20 von Commodore angefangen – und ich kann es seitdem nicht lassen, einen Computer zu reparieren oder etwas zu programmieren. Von dBase bis Access, von Basic bis VisualBasic und natürlich auch HTML, Pearl, PHP sowie ActionScript – der Programmiersprache in Macromedia Flash.

Das Spiel

Das Spiel, das ich für uns ausgesucht habe, hat folgendes Gameplay:

Du schwimmst mit unserem Held „Fishbud“ durch ein Aquarium. Vom oberen Rand fallen Fischfutterportionen ins Becken. Diese Portionen sollst du alle auffressen. Das Spiel ist vorbei, wenn du mehr als fünf Portionen verpasst hast.

Das Gameplay ergibt das Drehbuch

Bevor wir loslegen, müssen wir uns zuerst Gedanken über die allgemeine Reihenfolge machen. **Flash arbeitet wie ein digitales Daumenkino.** Alle Aktionen müssen wir auf einer Zeitleiste planen, die später automatisch abläuft – eben wie ein Daumenkino, wo einzelne Blätter nacheinander ablaufen, um in ihrer Gesamtheit eine Animation zu ergeben.

Deswegen ist für alle Gestaltungen und Programmierungen in Flash ein eigenes Drehbuch wichtig. Eine grobe Storyrichtung sollte von Beginn an feststehen; während der Programmierung sollte aus dieser groben Formulierung eine verfeinerte werden, z.B. um Variablen und Programmierstrukturen genau abzubilden. Zu Beginn sieht unser Drehbuch so aus:

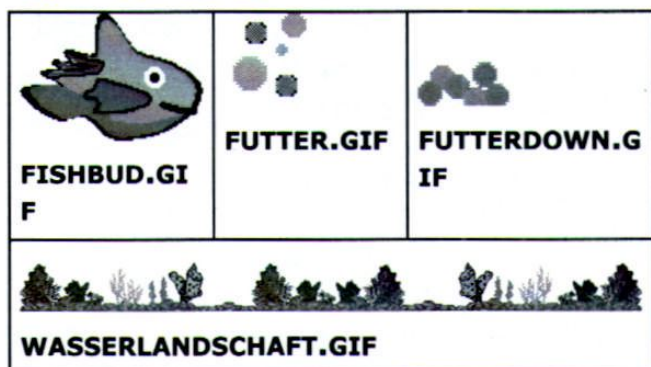
- (1) Fishbud schwimmt in einem Aquarium. Er reagiert auf Tastatureingaben und bewegt sich entsprechend nach links, rechts, oben und unten – die sensiblen Tasten sollen die Pfeiltasten sein, und beim Drücken der Leertaste soll Fishbud das Maul öffnen und fressen.
- (2) Es fallen in regelmäßigen Abständen Futterportionen von oben herab, die Fishbud fressen soll/kann.
- (3) Haben mehr als fünf Portionen des Futters den Grund erreicht, ist das Spiel zu Ende. Es wird eine Menüseite aufgerufen, von der aus der Spieler zum Spielstart zurück gelangt oder an einem Gewinnspiel teilnehmen kann. Optional kommt hier der Aufruf einer Highscoreliste

D.h. wir haben insgesamt drei „Hauptakte“ in unserem Drehbuch, die wir nun nach und nach abarbeiten und verfeinern. Es ist sehr hilfreich, wenn du dir z.B. in Word dieses Drehbuch aufbaust; später wird es dir von großem Nutzen sein, vor allem bei dem Umgang mit den verschiedenen Variablen und ACTIONSCRIPTS().

Die notwendigen Grafiken:

Zu Beginn unserer Spielprogrammierung sollten die „Hauptgrafiken“ vorhanden sein. Wir brauchen eine Grafik für unseren Helden Fishbud, eine Grafik für die Futterportionen und evtl. weitere Grafiken für die Spielumgebung, z.B. einen Meeresgrund.

Diese Grafiken bekommst du z.B. im Internet, oder du arbeitest selbst welche aus. Ich habe folgende vier Grafiken mit PhotoImpact von Ulead gestaltet und als GIF abgespeichert. Diese Bilder stehen auf www.spieleinflash.de zum Download bereit.



FUTTER.GIF ist ein animiertes Gif, das die diversen Kreissymbole in verschiedenen Farben blinken lässt.

FUTTERDOWN.GIF brauchen wir für den Fall, dass eine Portion den Boden erreicht.

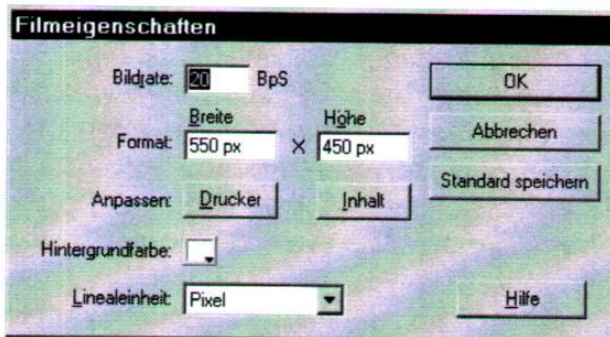
FISHBUD.GIF ist bisher noch nicht animiert, da die Animation der Flossen mit Flash viel professioneller aussieht als ein animiertes Gif – aber dazu später mehr. Bevor es in Flash los geht, legen wir für unser Spiel am besten ein eigenes Verzeichnis an, das ich **C:\FISHBUD** nenne – hier werden zunächst unsere Grafiken abgelegt.

Los geht's

Wenn du Flash startest, öffnet das Programm automatisch den Film **FILM1.FLA**.

Alle Dateien von Flash heißen *FILM*. In vielen Dokumentation taucht deswegen die Abkürzung MC auf – dies steht für **M**ovie**C**lip und meint die normale Dateiarart von Flash.

Bevor wir richtig loslegen, muss noch etwas in den Grundeinstellungen für den Film geändert werden. Über **MODIFIZIEREN|FILM** gelangst du in das folgende Dialogfeld:



Standardmäßig ist die Bildrate je Sekunde auf 12 eingestellt. Ich empfehle für unser Spiel eine Rate von 20 Bildern. Die Animationen und Bewegungen laufen dann erheblich besser ab. Nun passen wir noch die Höhe des Filmes auf 450 an. Per Klick auf OK wird dieses Fenster geschlossen und die geänderten Einstellungen übernommen.

Der 1. Akt

Bleiben wir beim Beispiel „Daumenkino“. Ein Daumenkino braucht für den reibungslosen Ablauf einzelne Bilder. Für Flash sind diese Bilder die sogenannten Frames. Die Anzahl dieser Frames wird auf einer Zahlenleiste dargestellt.

Hier kannst du sozusagen sekundengenau deine Animationen und Abläufe planen; dabei werden die Zeitabschnitte als Balken dargestellt, so dass beim Arrangieren deiner Objekte das ganze wie ein Tetrispiel aussieht.

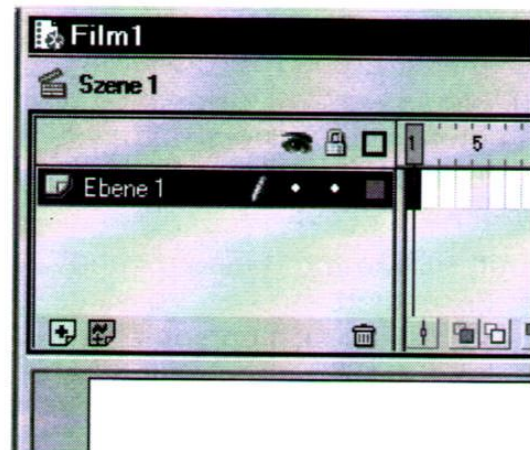
Unterhalb der Zeitleiste hast du mehrere Ebenen. In jeder Ebene verwaltest du einzelne Objekte bzw.

Symbole. Fishbud kommt in eine Ebene, das Futter kommt in eine weitere Ebene, die Programmierungen kommen wiederum in eine Ebene usw.

Willst du nun eines der Objekte animieren, soll also z.B. das Fischfutter später vom oberen Rand bis zum unteren Rand fallen, musst du Flash die „Haltepunkte“ dieser Animation als sogenannte Schlüsselbilder angeben.

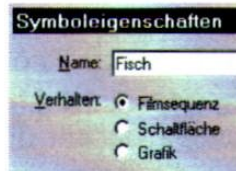
Im ersten Schlüsselbild liegt das Futter am oberen Rand; im zweiten Schlüsselbild am unteren. Je weiter du das zweite Schlüsselbild auf der Zeitleiste nach hinten schiebst, desto länger braucht das Futter, um zu fallen.

Wie das genau geht, erfährst du später. Zunächst ist wichtig, dass du jedes Objekt mit Schlüsselbildern auf deiner Zeitleiste arrangieren musst. Diese Schlüsselbilder müssen in Flash manuell angelegt werden. Für das erste Schlüsselbild klickst du einmal mit der linken Maustaste in *Ebene 1* auf den Frame 1 – also das erste Rechteck in dieser Zeitleiste.



Über **EINFÜGEN|SCHLÜSSELBILD** fügst du an dieser Stelle ein Schlüsselbild ein. Als nächstes ist die Grafik **FISHBUD.GIF** an der Reihe. Hierzu musst du wissen, dass man alle Objekte, die man in Flash animieren will, in sogenannte Symbole umwandeln muss. Bevor du also die Grafik **FISHBUD.GIF** einfügst, legst du zuerst ein solches Symbol an. Hierzu klickst du auf **EINFÜGEN|NEUES SYMBOL**. Als **NAMEN** wählst du hierfür *Fisch* – als **VERHALTEN** wählst du **FILMSEQUENZ**

Das VERHALTEN FILMSEQUENZ ist die Standardeinstellung von Flash, die z.B. gegenüber dem Grafiksymbol folgende Vorteile hat:

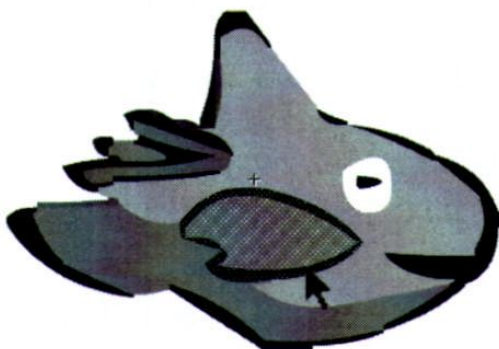


- Sie kann auch weitere Symbole enthalten
- Du kannst Sounds hinterlegen
- Du kannst eigene Aktionen hinterlegen

Als nächstes kannst du nun die Grafik importieren – und zwar mit DATEI|IMPORTIEREN. Hast du das Bild importiert, kannst du die Grafikqualität von Fishbud verbessern, indem du das Bild in eine Vektorgrafik umwandelst. Dazu klickst du es an und wählst MODIFIZIEREN|BITMAP NACHZEICHNEN.

Die vorgeschlagenen Einstellungen kannst du übernehmen. Danach solltest du die einzelnen Objekte von Fishbud wieder zu einem Ganzen kombinieren. Dazu markierst du mit dem UNTERAUSWAHL-WERKZEUG die neue Vektorgrafik und gruppierst sie über MODIFIZIEREN|GRUPPIEREN.

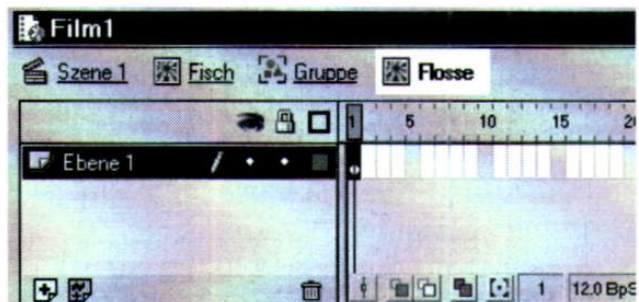
Als nächstes animierst du die Flosse von Fishbud. Um der besseren Übersicht willen wählst du am besten ANSICHT|VERGRÖßERN. Da du nun mehrere Teilbereiche markieren willst, hältst du die UMSCH-TASTE gedrückt, während du mit dem PFEILWERKZEUG die Flosse und ihre schwarze Umrandung auswählst.



Hast du alle Bereiche der Flosse markiert, gruppierst du diese über MODIFIZIEREN|GRUPPIEREN.

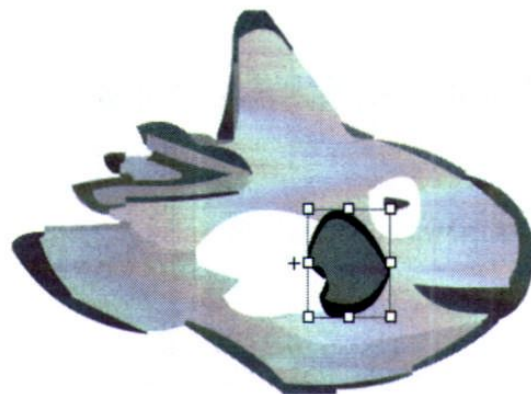
Willst du die Flosse animieren, musst du sie in ein Symbol umwandeln. Da die Flosse nach der Gruppierung noch immer markiert ist, kannst du dies direkt machen, indem du im Menü EINFÜGEN|IN SYMBOL KONVERTIEREN wählst. Als Name gibst du **FLOSSE** an.

Klickst du nun mit der Maus die Flosse doppelt an, gelangst du in die Zeitleiste für dieses Element.



Da du die Flosse animieren willst, benötigst du neue Schlüsselbilder. Hierfür kannst du in der Zeitleiste die einzelnen Frames anklicken. Du klickst z.B. auf den fünften Frame und fügst per rechter Maustaste SCHLÜSSELBILD EINFÜGEN ein neues Schlüsselbild ein.

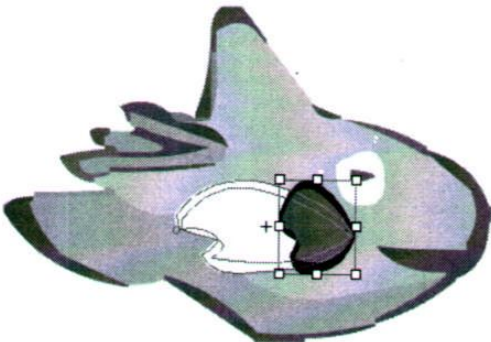
Hier kannst du nun die Veränderung der Flosse vornehmen. Über MODIFIZIEREN|TRANSFORMIEREN|SKALIEREN aktivierst du den Skalierrahmen. So kannst du die Flosse in ihrer Form verändern. Dazu ziehst du den Rahmen mit der Maus so, dass es aussieht, als wäre die Flosse hochgezogen.



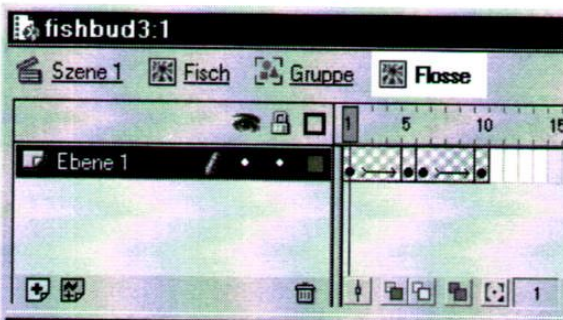
Flash braucht für jede Animation immer ein Schlüsselbild am Anfang und ein Schlüsselbild am Ende der Animationsphase.

In Frame 1 ist die Flosse ganz zu sehen – hier ist Schlüsselbild1; in Frame 5 ist die Flosse nun „hochgezogen“ – hier ist Schlüsselbild2. Für die Rückstoßbewegung brauchst du nun zwei weitere Schlüsselbilder – und zwar eines für den Beginn der Rückstoßbewegung und eines für ihr Ende.

Du brauchst also je ein weiteres Schlüsselbild in Frame Nr. 6 und in Frame Nr. 10. Hierfür klickst du in der Zeitleiste auf Frame Nr. 6 und fügst hier wieder per rechter Maustaste **SCHLÜSSELBILD EINFÜGEN** ein neues Schlüsselbild ein. Das gleiche machst du auf Frame 10. Nun kannst du die Flosse in Frame 10 über **MODIFIZIEREN|SKALIEREN** in die alte Form bringen.



Als letzten Schritt klickst mit der rechten Maustaste du auf den „grauen“ Zwischenraum zwischen Frame 1- und 5 und wählst **BEWEGUNGS-TWEEN ERSTELLEN** – ditto im Zwischenraum zwischen Frame 6 und 10! Fertig!



Um die weiße Fläche unter Flosse weg zu bekommen, mußt du zurück in den Bearbeitungsmodus für Fisch.

Hierfür klickst du oben in der Zeitleiste einfach auf Fisch. Hierin ziehst du mit der Maus die Flosse nach oben. Nun klickst du mit dem **PIPETTE-WERKZEUG** auf die grüne Fläche im Fisch. Dann klickst du mit dem direkt erscheinendem **EIMER** auf die weiße Fläche.

Du mußt danach die Flosse wieder an ihren Platz schieben.



Mit einem Klick auf **SZENE1** gelangst du nun zurück zur Hauptbühne. Um zu sehen, wie der „schwimmende“ Fishbud aussieht, mußt du ihn nur noch auf die Bühne bringen.

Hierzu öffnest du die Bibliothek der Objekte, die dir in diesem Film zur Verfügung stehen – und zwar per **FENSTER|BIBLIOTHEK**. Mit der Maus kannst du das Symbol „Fisch“ auf die Bühne ziehen; per **STEUERUNG|FILM TESTEN** startet der Film, und du siehst, wie der „schwimmende“ Fishbud ausschaut.

Blinkt Fishbud wie wahnsinnig, hast du in der Zeitleiste wahrscheinlich hinter dem ersten Schlüsselbild ein leeres Schlüsselbild. In diesem Falle klickst du einfach mit der Maus dieses Schlüsselbild an und löschst es mit der **ENTF-TASTE** – dann klappt es.

Diesen Stand **FISHBUD1.FLA** kannst du auf www.spieleinflash.de herunterladen.

Die Steuerung mit der Tastatur

Während man in Flash 4, für die Steuerung von Objekten, noch mit versteckten Schaltflächen arbeiten musste, kann man ab Flash 5 den Filmsequenzen die Tastaturereignisse direkt

zuweisen. Doch zunächst sollte Fishbud einen Instanznamen bekommen. Hierfür klickst du im Fenster **INSTANZ** auf den Reiter **INSTANZ** und gibst unter **NAME** *Fisch* ein



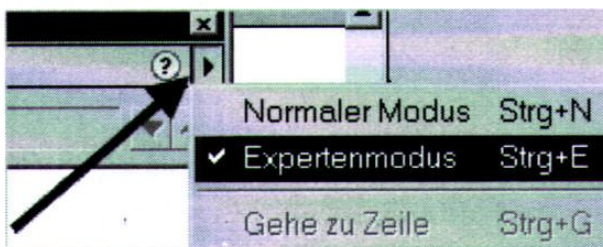
Ist dieses Fenster nicht geöffnet, klickst du mit der rechten Maustaste auf Fishbud und wählst **BEDIENFELDER|INSTANZ**.

Nachdem du den Namen eingetippt hast, solltest du RETURN drücken, da Flash andernfalls dazu neigt, den Namen zu ignorieren.

Der INSTANZNAME ist für die Programmierung in Flash absolut wichtig. Willst du später die Objekte ansprechen, benötigen sie eigene Namen, um z.B. Verwechslungen auszuschließen. Gleichzeitig behältst du in der Programmierung eine bessere Orientierung.

Um in den ACTIONSCRIPT() Modus zu gelangen, klickst du mit der rechten Maustaste auf Fishbud und wählst im erscheinendem Menu **AKTIONEN** aus.

Es öffnet sich das Dialogfeld **OBJEKTAKTIONEN**. An der rechten oberen Kante öffnet sich nach einem Klick auf den rechtsgerichteten Pfeil direkt unter der Schließfläche des Dialogfeldes ein Kontextmenü. Hier wählst du nun den **EXPERTENMODUS** aus, denn andernfalls kannst du im Aktionsfenster nicht drauf los tippen.



Es gibt in ACTIONSCRIPT den Befehl `onClipEvent(keydown) {...Befehle...}`, Benutzt du diesen Befehl, werden im Falle eines Tastendrucks die Befehle ausgeführt, die in den geschweiften Klammern stehen. Hier kann man z.B. abfragen, welche Taste gedrückt wurde.

Jede Taste hat ihren eigenen Keycode, d.h. die Tastatur liefert einen bestimmten ASCII-Code für jede Taste. Eine komplette Liste dieser Werte findest du auf www.spieleinflash.de

Du benötigst für die Steuerung die Werte der Pfeiltasten. Diese sind Links=37, Rechts=39, Oben=38, Unten=40, Leertaste=32.

Die logische Programmierfolge lautet:

```
Wenn eine Taste gedrückt wurde
dann {
  wenn die Taste 37 gedrückt wurde {
    dann verschiebe "Fisch" um
    ...Pixel nach links}
  wenn die Taste 39 gedrückt wurde {
    dann verschiebe "Fisch" um ...
    Pixel nach rechts}
  wenn die Taste 38 gedrückt wurde {
    dann verschiebe "Fisch" um ...
    Pixel nach oben}
  wenn die Taste 40 gedrückt wurde {
    dann verschiebe "Fisch" um ...
    Pixel nach unten}
}
```

Dies wird in ACTIONSCRIPT mit bekannten IF-Abfragen gemacht – und so sieht das ganze dann aus:

```
onClipEvent (keyDown) {
  // nach oben
  if (Key.getCode() == 38) {
    _level0.Fisch._y -= 3;
  }
  // nach unten
  if (Key.getCode() == 40) {
    _level0.Fisch._y += 3;
  }
  // nach links
  if (Key.getCode() == 37) {
    _level0.Fisch._x -= 3;
  }
  // nach rechts
  if (Key.getCode() == 39) {
    _level0.Fisch._x += 3;
  }
}
```

Mit `_level0.Fisch._x` oder `_y` sprichst du die Instanz **Fisch** auf **Ebene0** an – d.h. unseren Fishbud. X und Y verhalten sich wie im normalen Koordinatensystem. Probiere es einfach mal aus – nun lässt sich Fishbud schon lenken, zumindest was die Bewegung angeht.

Übrigens habe ich hier die Änderung der Position um je 3 Pixel vorgegeben. Natürlich kannst du den Wert noch ändern.

Fällt dir etwas auf? Richtig – wenn du nach links lenkst, sollte sich Fishbud in die andere Richtung drehen, und wenn später noch das Fressen hinzukommt, müsste er sich nach oben lehnen und das Maul öffnen. Es muss sich also bei bestimmten Tasten nicht nur die Position ändern, sondern auch das Aussehen von Fishbud.

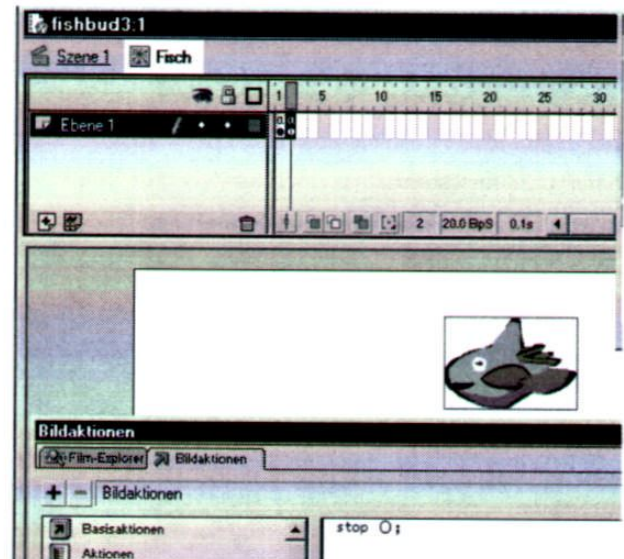
Grafikwechsel durch Tastaturklick

Mit einem Doppelklick auf Fishbud gelangst du in die Zeitleiste von Fishbud. Im Moment gibt es hier nur ein einziges Schlüsselbild in Frame 1. Um weitere Ansichten von Fishbud zu bekommen, musst du nun weitere Schlüsselbilder einfügen. Hierfür klickst du den Frame 2 mit der rechten Maustaste an und wählst **SCHLÜSSELBILD EINFÜGEN**.

Da der neue Fishbud schon markiert ist, kannst du direkt weitermachen und über **MODIFIZIEREN|TRANSFORMIEREN|HORIZONTAL SPIEGELN** diesen Fishbud nach „links“ schwimmen lassen.

Da Flash alle Zeitleisten automatisch wie ein Daumenkino ablaufen lässt, musst du in beiden Schlüsselbildern die **ACTIONSCRIPT `Aktion Stop()`** einsetzen.

Hierfür klickst du Frame 1 mit der rechten Maustaste an und wählst **AKTIONEN**. Nun kannst du im erscheinenden Fenster im linken Bereich den Befehl **`stop()`**; doppelt anklicken, oder du gibst ihn im rechten Fensterbereich selber ein.



Vergiss nicht, über den Pfeil oben rechts den **EXPERTENMODUS** einzustellen.

Ebenso verfahrst du auf Frame 2.

Übrigens – wenn du auf einen Frame ein Script gelegt hast, wird dir das durch das **a-Symbol** angezeigt.

Nun gehst du zurück in die Szene 1. Hier klickst du mit der rechten Maustaste auf Fishbud und wählst **AKTIONEN**. In die **`onClipEvent(Keydown)`**-Programmierung muss nun noch folgendes hinein:

Wenn die Rechtspfeil-Taste gedrückt wird { dann rufe den Frame1 in Fisch auf } wenn die Linkspfeil-Taste gedrückt wird {dann rufe den Frame2 in Fisch auf}

Die Abfragen „Wenn die Taste...“ sind schon da, d.h. dass nur der Aufruf der Frames eingefügt werden muss.

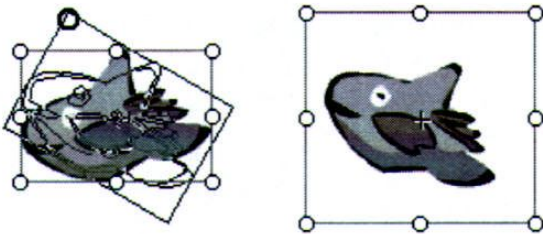
```
if (Key.getCode() == 37) {
    _level0.Fisch.gotoAndStop(2);
    _level0.Fisch._x -= 3;
}
```

Mit `_level0.Fisch` sprichst du die Instanz **Fisch** an, mit **`gotoAndStop(2)`** rufst du Frame 2 auf und damit den nach links schwimmenden Fishbud.

Entsprechend ergänzt du bei der Abfrage nach dem KeyCode 39 = rechts, nur hier gehst du eben nach Frame1. Fertig; nun schwimmt Fishbud auch in die richtige Richtung.

Als nächstes kommt das Fressen: Hier gehst du zunächst genauso vor wie beim Drehen nach links, d.h. du gehst in Fishbud und fügst ein weiteres Schlüsselbild ein. Auch hier gibst du als AKTIONEN **Stop()** ein.

Nun klickst du Fishbud an und wählst **MODIFIZIEREN|TRANSFORMIEREN|DREHEN** aus. Mit der Maus kannst du auf eine Ecke von Fishbud klicken und ihn per gedrückter Maustaste drehen.



Anschließend wählst du im Hauptmenü **MODIFIZIEREN|GRUPPIERUNG AUFHEBEN**.

Mit dem **PINSEL-WERKZEUG** malst du nun die schwarze Kontur des geöffneten Mauls. Das geht einfacher, wenn du vorher **ANSICHT|VERGRÖßERN** wählst. Nachdem die Kontur steht, wählst du die Farbe weiß als Farbe und malst das geöffnete Maul damit aus.



Danach gruppierst du alles wieder. Fertig! Nun hast du einen nach links schwimmenden, fressenden Fishbud.

In Frame 4 von Fishbud kommt nun der nach rechts schwimmende, fressende Fishbud. Hierfür fügst du in Fishbud auf Frame 4 ein neues Schlüsselbild ein. Da Frame 3 eins zu eins kopiert wird, brauchst du Fishbud per **MODIFIZIEREN|TRANSFORMIEREN|HORIZONTAL SPIEGELN** nur noch umdrehen.

Vergiss nicht das **Stop()** ; als **ACTIONSCRIPT-Befehl** unter **Aktionen**.

Nun muss in die Tastaturabfrage noch die Erweiterung um diese zwei fressenden Formen von Fishbud eingefügt werden. Zuerst klickst du oben links auf **SZENE 1**, um wieder in den Hauptfilm zu gelangen. Hier klickst du Fishbud mit der rechten Maustaste an und wählst **AKTIONEN**.

Hier muss nun ergänzt werden, dass im Falle eines Drucks auf die Leertaste einer der Frames 3 oder 4 aufgerufen wird. Frame 3 soll aufgerufen werden, wenn Fishbud nach links schwimmt und der Spieler die Leertaste drückt; Frame 4 dann, wenn Fishbud nach rechts schwimmt und der Spieler die Leertaste drückt.

Variablen = Merktzettel für wichtige Informationen

Dummerweise weiß Flash nicht, wohin Fishbud schwimmt; Flash kennt nur Frames und Schlüsselbilder aber keine logischen Zusammenhänge.

D.h. wir müssen uns in Flash ein eigenes „Gedächtnis“ anlegen. Hier notieren wir uns, wohin Fishbud gerade schwimmt.

Ähnlich wie ein Merktzettel, auf dem wir uns Dinge notieren, die wir nicht vergessen wollen. In der Programmierung heißen solche Merktzettel leider nicht Merktzettel, sondern **Variablen**. Z.B. könnte man eine Variable anlegen mit dem Namen **richtung**.

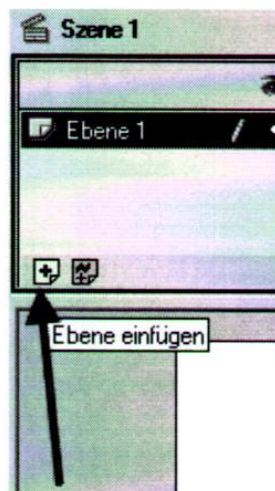
Zu Beginn des Spiels schwimmt Fishbud nach rechts, also ist der Startwert für diese **Variable** auch *rechts*. Wird nun die Taste für Links gedrückt, soll nicht nur der nach links schwimmende Fisch geladen werden, sondern es soll auch die Variable von **rechts** in **links** geändert werden.

Bevor man die Aktion für die Leertaste aufruft, schaut man nach, was auf dem Merktzettel Richtung steht, also welchen Inhalt die **Variable richtung** hat, und ruft dann passend Frame 3 oder Frame 4 auf.

Es macht Sinn, wenn du für Variablen und andere Programmierungen eine eigene Ebene hast.

So behältst du später eine bessere Übersicht. Eine neue Ebene fügst du per Klick auf das Symbol mit dem + ein, direkt unterhalb der Ebenenübersicht.

In dieser neuen Ebene klickst du mit der rechten Maustaste auf den Frame 1 und wählst **AKTIONEN** aus.



Am besten stellst du wieder den Expertenmodus ein.

Nun gibst du hier folgende Zeile ein:

```
var richtung="rechts";
```

Mit dieser Zeile hast du den "Merktzettel" *Richtung* geschrieben, auf ihm steht *rechts*. Nun klickst du mit der rechten Maustaste Fishbud an und wählst **AKTIONEN** aus. Bei den Abfragen für rechts und links ergänzst du nun die Änderung der **Variablen Richtung** mit folgenden Zeilen:

```
_root.richtung="links"; oder
```

```
_root.richtung="rechts";
```

Mit dem führenden **_root** weist du Flash an, im Hauptfilm diese Variable zu suchen und zu ändern. Wenn du dieses **_root** weg lässt, würdest du nur eine Variable generieren, die nur in dieser Ebene funktioniert!

Diese Zeilen kommen jeweils in die Schleifen für rechts und links. Nun kannst du sicher sein, dass du immer weißt, in welche Richtung Fishbud schwimmt. So kannst du nun die Abfrage für die Leertaste bauen.

Die logische Formulierung sagt:

```
wenn (die Leertaste gedrückt wird)  
wenn (Richtung=rechts)  
dann rufe Frame 4 auf  
wenn (Richtung=links)  
dann rufe Frame 3 auf.
```

Das Programm sieht dann so aus:

```
onClipEvent (keyDown) {
    // nach oben
    if (Key.getCode() == 38) {
        _level0.Fisch._y -= 3;
    }
    // nach unten
    if (Key.getCode() == 40) {
        _level0.Fisch._y += 3;
    }
    // nach links
    if (Key.getCode() == 37) {
        _level0.Fisch.gotoAndStop(2);
        _root.richtung="links";
        _level0.Fisch._x -= 3;
    }
    // nach rechts
    if (Key.getCode() == 39) {
        _root.richtung="rechts";
        _level0.Fisch.gotoAndStop(1);
        _level0.Fisch._x += 3;
    }
    // Leertaste
    if (Key.getCode() == 32) {
        if (_root.richtung=="rechts") {
            _level0.Fisch.gotoAndStop(4);
        }
        else {
            _level0.Fisch.gotoAndStop(3);
        }
    }
}
```

Die **Wenn-Formel** wird auch **Wenn-Dann-Sonst-Formel** genannt. Hier hast du im Augenblick eine **Wenn-Dann-Formel** gebaut. Da du hierbei nur zwei verschiedene Ergebnisse hast, ist folgende Formulierung kürzer und genauso effektiv:

```
if (Key.getCode() == 32) {
    if (_root.richtung=="rechts") {
        _level0.Fisch.gotoAndStop(4);
    }
    else {
        _level0.Fisch.gotoAndStop(3);
    }
}
```

Da die **Variable Richtung** nur zwei verschiedene Inhalte haben kann, ist klar, dass sie, wenn sie nicht **rechts** als Inhalt hat, nur **links** als Inhalt haben kann. Deswegen passt hier auch die **Wenn-Dann-Sonst-Formel!**

Die Grundfunktionen des 1. Aktes sind nun fertig. Fishbud reagiert auf die Pfeiltasten und die Leertaste und bewegt sich entsprechend.

In unserem Drehbuch kommt für den 1. Akt der Hinweis hinzu, dass wir mit einer Variablen arbeiten, die sich die Richtung von Fishbud merkt. Den Stand kannst du als **FISHBUD2.FLA** auf www.spieleinflash.de herunterladen

Exkurs: Steuerung in zwei Richtungen

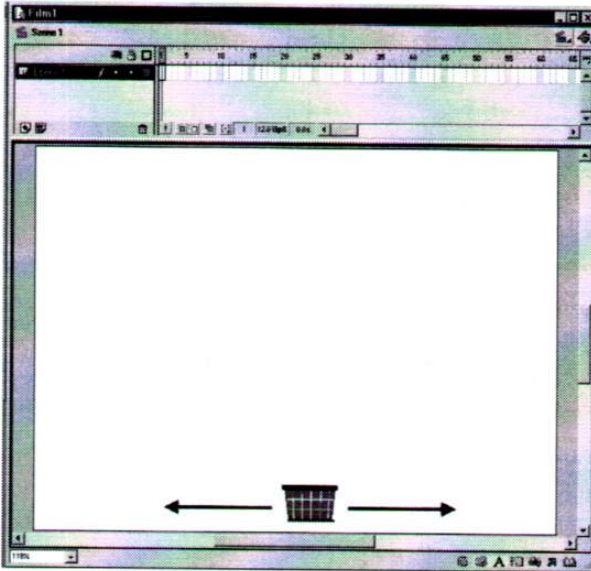
Es gibt Spielideen, in denen eine Bewegung in alle vier Richtungen nicht notwendig ist, z.B. beim „Eierfangen“. Hierbei hast du z.B. einen Korb, den du am unteren Bildschirmrand nach links und rechts bewegst.

Hierbei musst du lediglich beachten, dass du diesen Korb von Anfang an auf die richtige Höhe positionierst. Wie schon in Fishbud muss der Korb als Symbol vorliegen.

Entweder du importierst deine Grafik in ein Symbol oder du importierst die Grafik und wandelst sie über **EINFÜGEN|IN SYMBOL KONVERTIEREN** um.

Im Spiel ist die von dir gewählte Startposition zunächst absolut. Durch die Tastaturabfrage ergibt sich dann die Bewegung.

Gibt es keine **Key.getCode()**-Abfrage für die Tasten 38 (Pfeil nach oben) und 40 (Pfeil nach unten), bewegt sich das Objekt auch nicht in diese Richtung!



Als ACTIONSCRIPT benötigst du das gleiche Script wie bei Fishbud – die **IF-Abfragen** für die Richtungen, die du nicht brauchst, lässt du einfach weg.

Wahrscheinlich ändert sich bei diesem Spiel auch nicht das Erscheinungsbild des Korbes, so dass du diese Befehle ebenfalls weg lassen kannst.

```
if (Key.getCode() == 37) {
  _level0.Korb._x -= 3;}
if (Key.getCode() == 39) {
  _level0.Korb._x += 3;}
```

Exkurs: Steuerung mit der Maus

Grundsätzlich gehst du hierbei genauso vor wie bei der Steuerung mit der Tastatur. Zunächst musst du ein Objekt haben, das du steuern willst, z.B. den eben benutzten Korb. Als ACTIONSCRIPT() hinterlegst du ein `onClipEvent(load)`-Script.

In diesem Script schaltest du mit dem Befehl `Mouse.hide()`; den Mauszeiger aus, und mit dem Befehl `this.StartDrag(true)` weist du deinem Objekt die Mauskoordinaten zu. So sieht das Script aus:

```
onClipEvent(load) {
  Mouse.hide();
  this.startDrag(true);
}
```

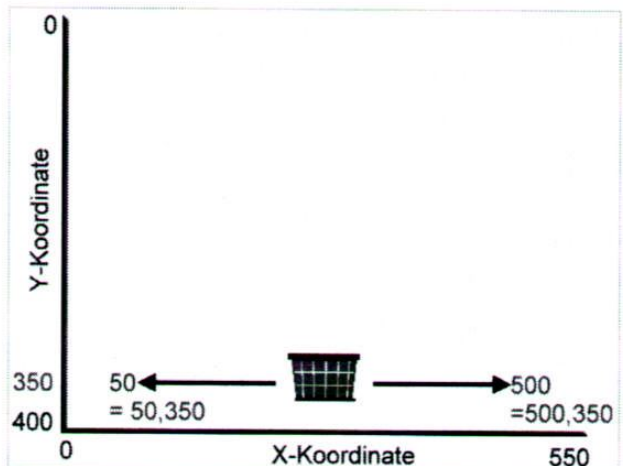
Willst du dein Objekt nicht in alle Richtungen bewegen, wie z.B. beim Eierfangen, kannst du den Bereich der Bewegung eingrenzen. Hierbei arbeitest du wieder mit XY-Koordinaten.

Darf sich dein Objekt z.B. nur horizontal bewegen, weist du die Koordinaten einer Linie zu. Diese Koordinaten kannst du in dem Befehl `startDrag()` gleich mit angeben.

Soll sich der Korb z.B. nur am unteren Rand bewegen, gibst du als erstes den Startpunkt der Linie an und zwar mit der XY Koordinate `50,350`. Enden soll die Linie dann auf `500,350`. `350` ist y-Koordinate und gibt die Höhe des Objektes an.

Die X – Koordinate hat einen Bereich von `50` bis `500` Pixel. Und so sieht dann das Script aus:

```
onClipEvent(load) {
  Mouse.hide();
  this.startDrag(true, 50, 350, 500, 350);
}
```



Willst du einen dreidimensionalen Raum als Bewegungsfläche begrenzen, änderst du einfach die Koordinaten, z.B. in

```
this.startDrag(true, 1, 1, 500, 500);
```

Die Kollisionskontrolle wird erst dann beschrieben, wenn wir die Objekte dafür haben. Im Beispiel von Fishbud ist das das Fischfutter – das als nächstes auf dem Programm steht.

Der 2. Akt

Im zweiten Akt unseres Drehbuches geht es um das Fischfutter. Dies soll in regelmäßigen Abständen von oben herabfallen. Hierfür benötigst du zunächst ein neues Symbol, das du über **EINFÜGEN|NEUES SYMBOL** einfügst. Das neue Symbol nennst du *Futter*, als **VERHALTEN** ist **FILMSEQUENZ** bereits richtig vorgegeben.

Flash öffnet sofort das Bearbeitungsfenster für das neue Symbol. Hier kannst du die vorhandene Futtergrafik importieren. Über **DATEI|IMPORTIEREN** gelangst du in die Dateiauswahl. Meine Vorlage ist ein animiertes Gif, das aus drei einzelnen Bildern besteht. Jedes dieser Bilder solltest du in eine Vektorgrafik umwandeln.

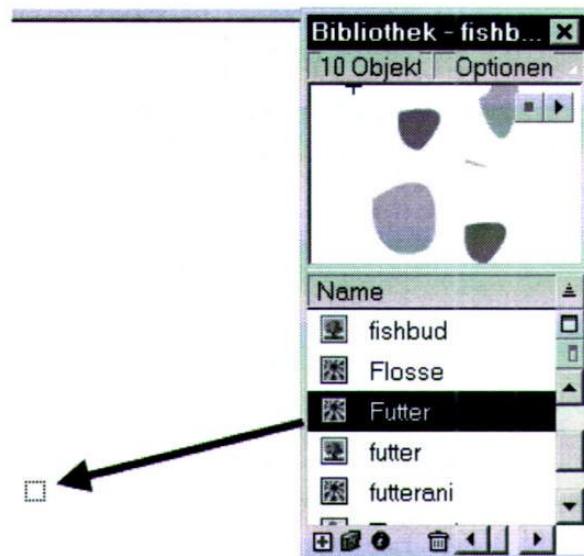
Da Flash automatisch die einzelnen Bilder eines **ANIMIERTEN GIF** erkennt, legt es für jedes einzelne Bild ein eigenes Schlüsselbild an. Für die Vektorisierung musst du nun jedes dieser Bilder separat umwandeln. Hierfür klickst du nacheinander die einzelnen Schlüsselbilder an und wählst **MODIFIZIEREN|BITMAP NACHZEICHNEN**.

Das Futter wird animiert

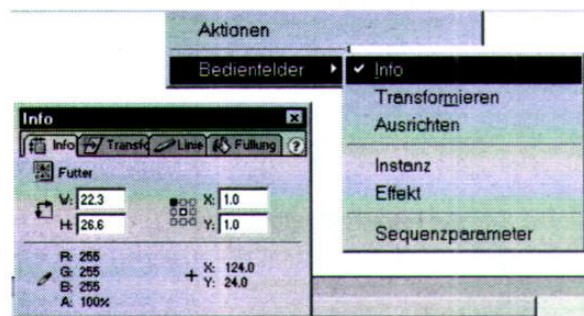
Als nächstes soll dieses *Futter* so animiert werden, dass es am oberen Filmrand auftaucht, sich langsam nach unten bewegt und unten schließlich als „Futterdown“ endet. Im Augenblick hast du ja bereits ein Symbol mit dem animierten Futter.

Für die Fallanimation brauchst du nun ein weiteres Symbol. Diese fügst du per **EINFÜGEN|NEUES SYMBOL** ein; als **NAME** vergibst du *Futterani* und als **VERHALTEN** wählst du **FILMSEQUENZ**.

In dieses Symbol fügst du das Symbol *Futter* ein. Über **FENSTER|BIBLIOTHEK** öffnest du ein Übersichtsfenster mit allen Symbolen und Grafiken, die dir zur Verfügung stehen. Hier wählst du *Futter* per Drag and Drop aus – d.h. du klickst *Futter* mit der linken Maustaste an, hältst die Maustaste gedrückt und bewegst die Maus auf die Arbeitsfläche.



Wenn du die Maustaste loslässt, ist *Futter* in *Futterani* eingefügt. Nun positionierst du das *Futter*, am besten per Menü. Hierfür klickst du mit der rechten Maustaste auf das *Futter* und wählst **BEDIENFELDER|INFO** aus.



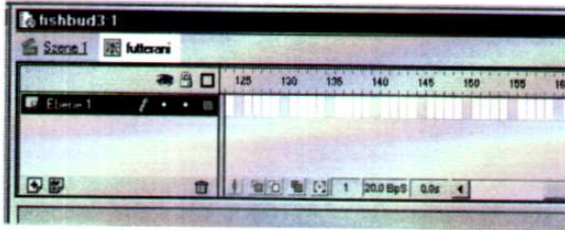
In diesem **INFO**-Fenster kannst du unter **X** und **Y** die Koordinaten für diese Grafik angeben, z.B. 1 und 1.

Nun kommt endlich einmal ein echtes Daumenkino:

In der Zeitleiste gibt es eine Einteilung in Frames. Diese Frames sind die Bilder, die nacheinander angezeigt werden. Für die Geschwindigkeit ist u.a. die Angabe unter **Bps** maßgebend (Bilder pro Sekunde).

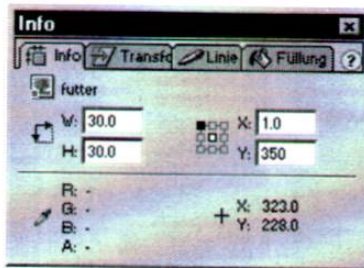
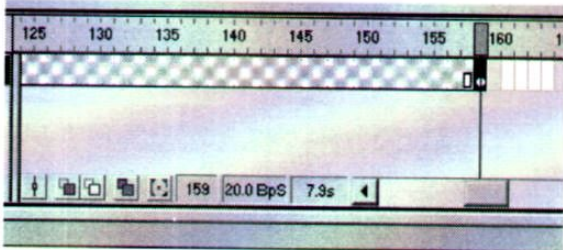
Als du den Film angelegt hast, hast du unter **MODIFIZIEREN|FILM** den Wert 20 hierfür eingegeben, d.h. pro Sekunde werden 20 Frames abgespielt. Wenn das *Futter* z.B. 8 Sekunden brauchen soll, um zum Grund zu gelangen, dann benötigst du $20 \times 8 \text{ Frames} = 160$. D.h. du musst eine Animation des *Futters* einfügen, die bis zum Frame 160 geht. Hierfür scrollst du mit der Leiste auf der

Zeitleiste soweit, bis zu bei Frame 160 angekommen bist.



Hier klickst du mit der rechten Maustaste den Frame 159 an und wählst **SCHLÜSSELBILD EINFÜGEN**. Dann klickst du dein *Futter* an und änderst die Koordinaten im INFO-Fenster. Für den Y-Wert gibst du 350 ein. Nun sieht die Sache so aus:

Das Futter in Frame 1 steht bei Position 1/1. Das Futter in Frame 159 steht bei Position 1/350.



In der Zeitleiste kannst du mit der rechten Maustaste auf den grauen Zeitbalken zwischen Frame 1 und Frame 159 klicken und per Menüpunkt **BEWEGUNGS-TWEEN ERSTELLEN** eine automatische Animation erstellen. Im Ergebnis verfärbt sich der Balken in lila und zeigt einen schwarzen Pfeil an.



Als nächstes klickst du mit der rechten Maustaste auf den Frame 160 und wählst **LEERES SCHLÜSSELBILD EINFÜGEN**. In dieses Schlüsselbild importierst du über **DATEI | IMPORTIEREN** die Grafik **FUTTERDOWN.GIF**.

Auch diese Grafik sollte per **MODIFIZIEREN | BITMAP NACHZEICHNEN** in eine Vektorgrafik umgewandelt werden.

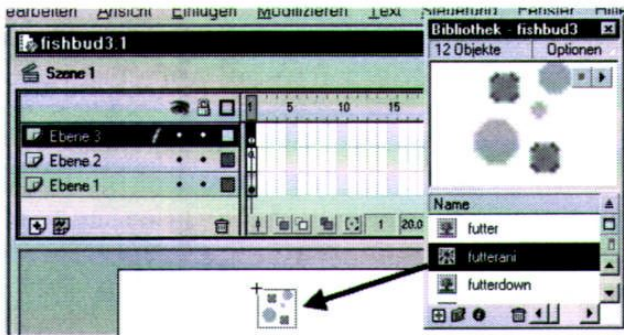
Nach der Vektorisierung klickst du *Futterdown* an und gibst im **INFO-FENSTER** als Position die Koordinaten 1 und 362 ein.

Da das fallende Futter ca. 12 Pixel „höher“ ist als das liegende Futter, muss der Y Wert gut 12 Pixel größer sein. Im Resultat liegt das „liegende“ Futter auch an der richtigen Stelle und nicht etwa zu hoch.

Flash würde diesen Film immer wieder von vorne abspielen, d.h. du musst in Frame 160 den **ACTIONSCRIPT** Befehl **stop()** einbauen, damit das Futter später nicht pausenlos nach unten fällt, ohne dass du einen Einfluss darauf hast.

Nun hast du ein Symbol mit dem Namen *Futterani*, in dem das Fischfutter von Position 1/1 innerhalb von 159 Einzelbildern bis zur Position 1/350 „herunterfällt“ und zum Schluss in Frame 160 das Aussehen in *Futterdown* ändert.

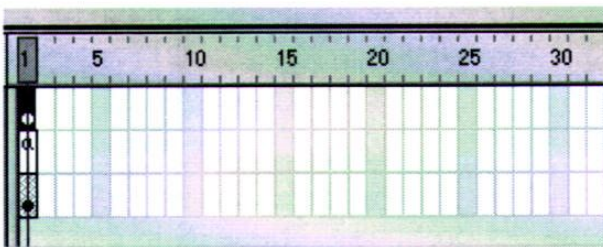
Du kehrst in den Hauptfilm zurück, indem du oben rechts auf **Szene 1** klickst. Hier fügst du dann eine Ebene für das Futter ein und ziehst aus der Bibliothek das Symbol *Futterani* auf die Filmfläche, z.B. an den oberen Rand.



Testest du nun den Film, schwimmt Fishbud umher, und das Futter fällt innerhalb von 8 Sekunden langsam zu Boden und bleibt dort liegen. Diesen Stand kannst du auf als **FISHBUD3.FLA** www.spieleinflash.de herunterladen.

Exkurs: So arrangiert man mehrere Objekte innerhalb der Zeitleiste

Bevor du mit der eigentlichen Arbeit beginnst, musst du dich entscheiden, wie du die Fischfutterportionen und Fishbud miteinander auf deiner Zeitleiste arrangierst. Basis für alle Objekte in unserem „Daumenkino“ ist die Zeitleiste im oberen Menü.



Die Zahleneinheit ist das Frame. Je nach Filmeinstellung spult Flash später diese Frames als Einzelbilder ab. In unserem Beispiel sind dies 20 Frames pro Sekunde. Von nun an musst du also neben der Programmierung und grafischen Bearbeitung deiner Objekte auch eine eigene Zeiteinteilung beachten.

Jedes Objekt, das du hier einbringst, hat entweder schon eine eigene Spieldauer – z.B. läuft das Futter 160 Frames –, oder es handelt sich um Objekte mit einer Programmierung wie z.B. Fishbud inkl. der Tastaturabfrage.

Stell dir vor, du spielst TETRIS und musst von nun an deine „Balken“ aneinanderreihen. Um mehrere Inhalte gleichzeitig zu sehen, nutzt du die Ebenen. Der Übersicht halber sollte jedes „Thema“ seine eigene Ebene

haben. D.h. du hast eine Ebene für Fishbud, eine Ebene für deine Programmierungen und eine weitere Ebene für das Futter. Später kommen Ebenen für den Punktstand, das Hintergrundbild etc. hinzu.

Im Augenblick hat der „Hauptfilm“ nur einen einzigen Frame. In diesen Frame hast du die Futteranimation eingefügt; diese wiederum hat eine eigene Spielänge von 160 Frames. Da sich diese beiden Zeitleisten noch nicht gestört haben, lief die Animation auch reibungslos.

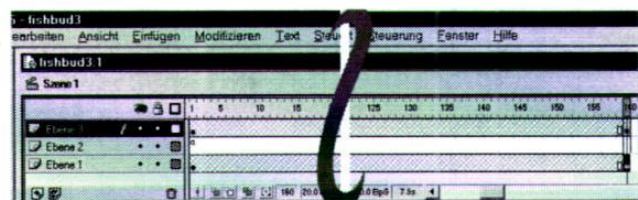
Da du aber nun die Zeiteinheiten in deinem Hauptfilm mit anderen Objekten arrangieren musst, musst du von nun an diese Laufzeit in der Hauptfilm-Zeitleiste berücksichtigen.

Das Futter läuft 160 Frames lang. Es ist im Augenblick in Frame 1 in der Ebene 3 eingefügt. Um deinen ersten „Tetrisbalken“ zu bekommen, klickst du mit der rechten Maustaste in der Zeitleiste auf den Frame 160 und wählst **SCHLÜSSELBILD EINFÜGEN**.

Nun siehst du, dass der Zwischenraum zwischen Frame 1 und Frame 160 grau unterlegt ist. Für Flash bleibt nun diese Grafik bis zum Erreichen des Frames 160 sichtbar.

Wenn du den Film testest, siehst du, dass das Futter zwar wie vorher herunterfällt, aber Fishbud ist nicht mehr zu sehen. Das liegt daran, dass Fishbud nur in Frame 1 als Schlüsselbild eingefügt ist – ab Frame 2 ist er also nicht mehr zu sehen.

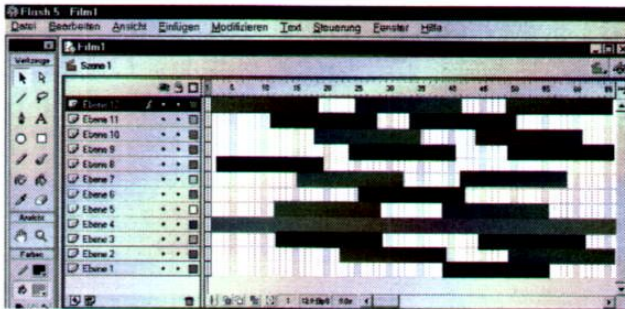
Soll Fishbud auch die ganzen 160 Frames zu sehen sein, musst du auch in der Ebene von Fishbud (Ebene 1) auf Frame 160 gehen und per **SCHLÜSSELBILD EINFÜGEN** die Anzeigedauer bis hier festlegen. Das Gleiche gilt für unsere Scriptebene.



Testest du nun den Film, ist alles wieder zu sehen. Da Flash beim Erreichen des Endframes automatisch wieder von vorne anfängt, fällt von nun an das Futter nicht nur einmal, sondern immer wieder von oben herab.

Laut Drehbuch sollen mehrere Portionen von oben herabfallen. Je länger das Spiel läuft, desto mehr Portionen sollen fallen. Das Spiel ist zu Ende, wenn fünf Portionen nicht gefressen wurden. Nun hast du zwei Möglichkeiten, dieses Gameplay umzusetzen:

In der ersten Möglichkeit kannst du für jede weitere Futterportion weitere Ebenen hinzufügen und an verschiedenen Framepunkten weitere Portionen von *Futterani* einfügen; am Ende könnte deine Zeitleiste dann so aussehen:



Eine weitere Möglichkeit ist das Erstellen von „Kopien“ des Futters während des Filmlaufes per `ACTIONSCRIPT()`. Dies hätte den Vorteil, dass du nur eine oder wenige Ebene(n) für die Futter-Portionen brauchst. Im Resultat wird die Arbeit auf der Zeitleiste übersichtlicher.

Zusammen mit `ACTIONSCRIPT()` ergeben sich zusätzliche Möglichkeiten, um das Gameplay aufregender zu gestalten – deshalb mache ich auch mit dieser Variante weiter.

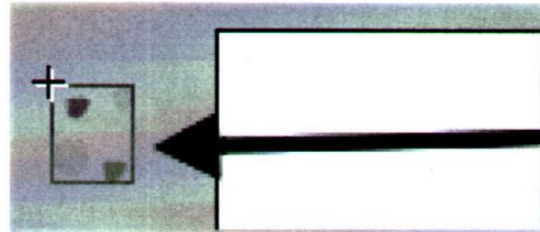
Du verpasst hierbei übrigens nichts, denn diese `ACTIONSCRIPT()` Variante beinhaltet auch das Aneinanderhängen von Filmsequenzen.

Klone des Futters erstellen

Sollen während des Spiels Klone bzw. Kopien von *Futterani* erstellt werden, musst du gewährleisten, dass sich das Originalsymbol *Futterani* ständig im Film befindet.

Flash kann nur Klone von einem im Film vorhandenem Symbol anfertigen.

Damit du später im Spiel nur mit den Klonen zu arbeiten brauchst und nicht mit dem Original, schiebst du das Symbol *Futterani* einfach an einen Platz außerhalb der Bühne.



Später im Spiel sind nur die Dinge sichtbar, die auf der Bühne liegen. Liegt dieses Originalfutter außerhalb der Bühne, ist es zwar ständig da – aber nicht zu sehen.

Dieses HAUPTFUTTER muss noch einen INSTANZNAMEN bekommen. Hierfür klickst du im Fenster INSTANZ auf den Reiter INSTANZ und gibst als NAME *Futter* ein. Mit diesem Instanznamen kannst du später in der Programmierung direkt auf dieses Symbol zugreifen.

Für die Programmierung der Klone nutzt du die Scriptebene – Ebene 2, in der du ja bereits die Variable *Richtung* eingebettet hast. Es sollen zu Beginn drei Portionen herabfallen. Diese Portionen sollen mit festgelegten Abständen voneinander entfernt sein.

Hierbei musst du übrigens genauso die Zeitleiste beachten wie vorher, d.h. wenn du willst, dass bei Frame 10 der erste Klon kommen soll, musst du auch in Frame 10 das ActionScript hierfür platzieren.

Letztendlich läuft diese Animation dann bis zum Frame 170, da das Futter 160 Frames zum Fallen braucht. Du musst dann jeden „Endframe“ deiner Ebenen von 160 auf 170 hinausziehen.

Hierzu klickst du die Endframes mit der linken Maustaste an, hältst die Taste gedrückt und ziehst dann die Maus auf den Frame 170. Wenn du dann die Maustaste loslässt, hast du den Endframe verschoben.

Du klickst in Ebene 2 den Frame 10 mit der rechten Maustaste an und wählst AKTIONEN. Vergiss nicht den Expertenmodus!

Du musst nicht unbedingt ausge-rechnet in Frame 10 beginnen; du kannst das auch in Frame 3 oder xx tun, nur nicht in Frame 2 – den Grund dafür wirst du später erkennen!

Bevor du in die Programmierung gehst, veranschauliche dir kurz das zu schreibende Programm:

```
Drei mal soll folgendes passieren
{
Erstelle einen Klon von Futter
(dies ist der Instanzname von
Futterani) und positioniere ihn an
einer bestimmten Stelle.
}
```

Da die geklonten Symbole dieselben Parameter haben wie ihr Original, steht jeder Klon auf der y-Koordinate 1. Für die Positionierung der Klone auf der x-Achse musst du nun ein wenig mit Mathematik arbeiten.

Z.B. kannst du eine Zufallszahl zwischen 0-10 generieren und diese dann mit 50 multiplizieren. Dies ergibt dann die x-Koordinate deines Futterklons. Somit erhältst du Futterportionen, die per Zufall von unterschiedlichen Positionen herabfallen, und machst damit das Spiel ein wenig interessanter.

Wenn du das nicht willst, kannst du dir die Zufallsgeschichte natürlich sparen und festgelegte Werte nehmen. Jetzt gehen wir allerdings den aufregenden Weg per Zufallszahl. Das zu schreibende Programm sieht so aus:

```
Drei mal soll folgendes Passieren
{
Ermittle eine Zufallszahl zwischen
0-10. Multipliziere diese Zahl mit
50 und speichere diese Zahl in der
Variablen mit dem Namen "x"
Erstelle einen Klon von Futterani
und positioniere ihn auf der
Position "x" }
```

Und so sieht dieses Programm aus:

```
setProperty ("futter", _visible, "0");
for (var i=0; i<3 ; i++) {
x = random(5);
x = x*100;
duplicateMovieClip
("futter","futter"+i, i);
setProperty ("futter"+i, _visible,
"1");
setProperty ("futter"+i, _x, 33+x);
}
```

In der ersten Zeile schaltest du pro forma die Sichtbarkeit unseres Futteroriginals aus. Zwar liegt es außerhalb der Bühne und ist nicht sichtbar –da die Grafiken beim Laden durch Flash aber manchmal aufblitzen, könnte diese Portion erscheinen. Mit dem „unsichtbar“-schalten verhinderst du das.

Danach kommt eine programmierte Schleife. Diese Schleife führt den Inhalt der Klammern solange aus, bis der Wert der Variablen **i** nicht mehr kleiner als 3 ist. Im Einzelnen:

- **var i=0;**
generiert eine Variable mit dem Namen **i** und dem Anfangswert =0
- **i<3;**
Dies ist die Bedingung, die erfüllt sein muss, damit der Inhalt der Klammern ausgeführt wird. Also muss **i** kleiner 3 sein.
- **i++**
hier wird **i** hoch gezählt, wenn die Klammer einmal durchgelaufen ist.
- **for (var i=0; i<3; i++)**
{Befehle}
In diesem Fall läuft die Klammer drei mal durch, genauso wie wir es wollen.

- `x= random(10);`
Ermittelt eine Zufallszahl im Bereich von 0-10 – `random(20)` Bereich von 0-20 etc.

Benötigst du eine Zufallszahl in einem Bereich ab 1 beginnend, z.B. bei einem Würfelspiel, zählst du nach der Ermittlung einfach eine 1 hinzu. So sähe dann das Script aus: `x=random(5); x=x+1;` – dies ergibt die **Variable x** mit einem Wert zwischen 1 und 6.

- `x=x*50;`
Hier multiplizierst die ermittelte Zufallszahl mit 50, um einen vernünftigen Wert für eine x-Koordinate zu erhalten. Unser Film ist 550 Pixel breit, für insgesamt 11 Startpositionen ist also genug Platz 0=kleinste x-Position, 500 = größte x-Position.
- `duplicateMovieClip ("futter", "futter"+i, i);`
Mit diesem Befehl klonst du das Futter.
Hier die Erläuterung:
`duplicateMovieClip (`
`"Originalname" , "Name des Klons", "Ebene des Klons")`
- `"futter"`
Hier sprichst du die Objekte mit ihren Instanznamen an; hätten wir unser *Futterani* also fischfutti genannt, müsste hier nicht „futter“ stehen, sondern „fischfutti“!
- `"futter"+i`
da unser Klon auch einen Instanznamen braucht, kombinierst du hier einfach das Wort `futter` mit dem `i`-Wert unserer Schleife von oben. Klon Nr. 1 heißt dann also `futter0`, Klon Nr. 2 heißt dann `futter1` und deine dritte Portion heißt `futter2`.

- `i`
Jedes Objekt muss auch eine eigene Ebene haben. Deshalb nutzt du hier auch die Variable `i` als Wert. Mit jedem Durchlauf wird `i` um eins hochgezählt und deshalb ist gewährleistet, dass jeder Klon auch seine eigene Ebene erhält.

Klone haben immer die gleichen Eigenschaften wie das Original. Da die Hauptgrafik in der ersten Zeile auf „unsichtbar“ gestellt wurde, sind die Klone zunächst auch unsichtbar.

- `setProperty ("futter"+i, _visible, "1");`
Mit dieser Zeile sprichst du den Klon an und weist seinem Attribut `visible` den Wert 1 zu.
0 heißt nicht sichtbar
1 heißt sichtbar.
Hier die Erläuterung:
`setProperty(Name des Objektes ,`
`Parameter der eingestellt werden soll , Wert für den Parameter)`
- `setProperty ("futter"+i, _x, 33+x);`
Hier wird der Klon positioniert. Da hier nur der `x`-Wert geändert wird, bleibt der `y`-Wert, den der Klon von der Hauptgrafik übernommen hat, unverändert auf 1.

Die Grafik ist 33 Pixel breit, deshalb musst du zum per Zufall ermittelten Wert für die x-Koordinate den Wert 33 hinzuzählen. Im Falle der Zufallszahl 0 wäre das Futter sonst außerhalb des sichtbaren Bereiches.

Testest du nun den Film, siehst du, dass per Zufall positionierte drei Portionen von oben herabfallen.

Kollisionskontrolle – Fishbud frißt das Futter

Nun schwimmt Fishbud laut Tasteingaben, und von oben fallen Fischfutterportionen herab. Damit Fishbud diese Portionen fressen kann, musst du eine sogenannte "Kollisionskontrolle" einbauen.

Diese überprüft, ob Fishbud beim „Fressen“ Kontakt zu einer der Futterportionen hat. Besteht ein Kontakt, kannst du diese Portion verschwinden lassen. Allerdings wäre es kein echter Spielspaß, wenn die Futterportionen auch dann „gefressen“ würden, wenn Fishbud sie mit seiner Hinterflosse berührt.

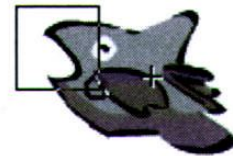
Das Futter darf nur dann gefressen werden, wenn Fishbud das Maul geöffnet hat, und wenn sich die Futterportion im „Maulbereich“ befindet. Ab Flash 5 gibt es hierfür den `ACTIONSCRIPT()` Befehl `hitTest`.

Dieser Befehl überprüft, ob sich zwei Symbole berühren, und liefert ein **JA** oder ein **NEIN** als Antwort. Die Futterportionen sind bereits Symbole. Nun muss also ein weiteres Symbol her, das den sensiblen Bereich bei Fishbud's Maul symbolisiert.

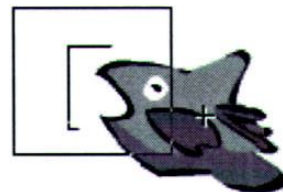
Zu Anfang hatte ich geschrieben, dass die Symbole, die als `VERHALTEN FILMSEQUENZ` haben, den Vorteil besitzen, dass man mehrere weitere Symbole in dieses Symbol integrieren kann.

Diesen Vorteil nutzt du nun für die Kollisionskontrolle. Du musst lediglich in die „fressenden“ Frames 3 und 4 von Fishbud ein Symbol einfügen, das den Bereich um Fishbud's Maul markiert.

Hierfür klickst du doppelt auf Fishbud. In der Zeitleiste wählst du den Frame 3 aus – den Frame, in dem Fishbud frisst. Nun wählst du das `RECHTECKWERKZEUG` aus und malst über den sensiblen Bereich von Fishbud's Maul ein Quadrat:



Direkt danach markierst etwas großzügiger das neue Quadrat mit dem `PFEILWERKZEUG`



und wandelst dieses Quadrat in ein Symbol um. Hierfür klickst du auf `EINFÜGEN|IN SYMBOL KONVERTIEREN`. Als `INSTANZNAMEN` gibst du hier `hitarea-links` ein. Da nach der Umwandlung in ein Symbol dieses Quadrat über Fishbud liegt und zum Teil die Ansicht verdeckt, musst du dieses Symbol nun noch auf `unsichtbar` schalten.



Hierfür klickst du mit der rechten Maustaste auf das Quadrat und wählst im Menü `AKTIONEN` aus.

Hier muss nun folgende Programmierlogik eingegeben werden:

Wenn dieser Frame geladen wird, dann soll die Sichtbarkeit auf "Falsch" gesetzt werden

```
In ActionScript sieht das dann so aus:
onClipEvent (enterFrame) {
  this._visible = false;
}
```

Testest du nun den Film, siehst du dieses Quadrat nicht – aber es ist da, und du kannst hiermit eine Kollisionskontrolle durchführen.

`this._visible=0`; hat den gleichen Effekt!

0 oder `false` = nicht sichtbar

1 oder `true` = sichtbar

Vergiss nicht, dass der gleiche Vorgang auch für den nach rechts fressenden Fishbud gemacht werden. Also auf Frame 4 klicken und hier ebenso ein Quadrat als Symbol einfügen – INSTANZNAME = `hitarearechts` – und das entsprechende ActionScript hinterlegen.

Der ACTIONSCRIPT-Befehl `hitTest` überprüft, ob sich zwei namentlich genannte Symbole berühren. D.h. du musst für jede vorkommende Futterportion eine eigene Abfrage machen! Die Programmierlogik stellt sich so dar:

Wenn dieser Frame geladen wird {
Wenn dieses Symbol mit Futter0 kollidiert

(lösche Futter0 aus dem Film und schließe das Maul von Fishbud)

Wenn dieses Symbol mit Futter1 kollidiert

(lösche Futter1 aus dem Film und schließe das Maul von Fishbud)

Wenn dieses Symbol mit Futter2 kollidiert

(lösche Futter2 aus dem Film und schließe das Maul von Fishbud)

}

Diese Programmierung kommt in die `onClipEvent(enterFrame)`-Schleife, die du für die Hitareas programmiert hast. D.h. unterhalb von `this._visible = false`; kommen nun die Abfragen der Kollisionen:

```
if (this.hitTest(_root.futter0)
{
removeMovieClip(_root.futter0)
_level0.Fisch.gotoAndPlay(2);
}
if (this.hitTest(_root.futter1)
```

```
{
removeMovieClip(_root.futter1)
_level0.Fisch.gotoAndPlay(2);
}
if (this.hitTest(_root.futter2)
{
removeMovieClip(_root.futter2)
_level0.Fisch.gotoAndPlay(2);
}
```

Die Ansprache `_root` ist eine absolute Adressierung innerhalb des Filmkonstruktes oder Strukturbaumes deines Filmes. `Futter0`, `Futter1` und `Futter2` sind die INSTANZNAMEN, die du beim Klonen direkt vergeben hast.

- `this`
das Symbol, auf dem dieses Script liegt, also das Rechteck über Fishbud's Maul
- `hitTest`
Kollisionskontrolle
- `this.hitTest(_root.futter2)`
prüft, ob dieses Symbol mit dem Symbol, das in der Klammer genannt ist, kollidiert
- `removeMovieClip(_root.futter0)`
löscht den Film, der in der Klammer genannt wird, aus diesem Film
- `_level0.Fisch.gotoAndPlay(2)`;
ruft den Frame 2 von Fishbud auf, in dem Fishbud mit geschlossenem Maul nach links schwimmt. Eigentlich sollte hier `gotoAndStop()` stehen; da du aber jeden Frame von Fishbud mit `stop()`; schon selbst stoppst, ist es hierbei egal, welche der beiden Varianten du nutzt.

Bei der Kollisionskontrolle des rechts fressenden Fishbud muss der Frame 1 aufgerufen werden, in dem Fishbud mit geschlossenem Maul nach rechts schwimmt

So sieht das Programm für das nach rechts geöffnete Maul aus:

```
onClipEvent (enterFrame) {
    this._visible = false;
    if (this.hitTest(_root.futter0)) {

removeMovieClip(_root.futter0);
        _root.gefressen++;
        _level0.Fisch.gotoAndPlay(1);
    }
    if (this.hitTest(_root.futter1)) {
        removeMovieClip(_root.futter1);
        _root.gefressen++;
        _level0.Fisch.gotoAndPlay(1);
    }
    if (this.hitTest(_root.futter2)) {
        removeMovieClip(_root.futter2);
        _root.gefressen++;
        _level0.Fisch.gotoAndPlay(1);
    }
}
```

Damit ist der 2.Akt fertig! Nun fallen in regelmäßigen Abständen Futterportionen von oben herab, die Fishbud fressen kann. Im Drehbuch muss nun eingefügt werden, dass man eine Originalportion des Fischfutters einfügt, diese unsichtbar schaltet und von dieser Portion Klone für das Spiel generiert.

Diesen Stand kannst du als

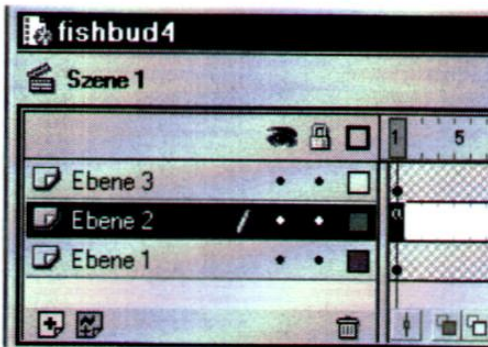
FISHBUD4.FLA auf

www.spieleinflash.de herunterladen.

Der 3. Akt

Nun geht es darum, dass sowohl die gefressenen als auch die verlorenen Portionen mitgezählt werden müssen. Wenn Fishbud mehr als fünf Portionen verloren hat, ist das Spiel beendet. Sehen wir uns wieder die vorher beschriebenen Variablen an, also die Merktzettel der Programmierung.

Sie eignen sich hervorragend für das Mitzählen! Mit anderen Worten brauchst du nun zwei weitere Variablen. Hierfür klickst du in der Ebene 2 auf Frame 1.



Hier hast du ja bereits einen Merktzettel für die Richtung von Fishbud programmiert. Nun kommen hier zwei weitere Merktzettel hinzu, also zwei weitere Variablen:

gefressen = gefressene Portionen und
verloren = verlorene Portionen:

```
var richtung = "rechts";
var gefressen=0;
var verloren =0;
```

Der Startwert beider Variablen ist natürlich 0. Als nächstes ist zu überlegen, an welchen Stellen im Spiel die Punktezahlung erfolgen soll.

Die Variable **verloren** muss um einen nach oben gezählt werden, wenn der Frame 160 von *Futterani* = **FUTTERDOWN.GIF** geladen wird.

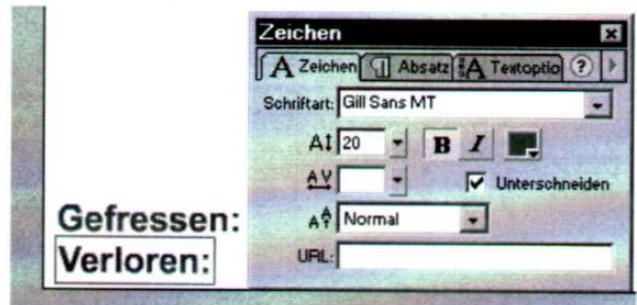
Der Punktstand für **gefressen** soll hochgezählt werden, wenn es eine Kollision zwischen einer der **Hitareas** und einer Futterportion gab.

Doch bevor du mit dem Punkte zählen beginnen kannst, sollten Anzeigen dieser Variablen vorhanden sein. So kannst du am Monitor besser verfolgen, ob die Variablen auch richtig hochgezählt werden.

Exkurs: Punkteanzeige als Textfeld

Für die Textfelder und Anzeigen im Film brauchst du natürlich auch eine eigene Ebene: Hier klickst du auf den Frame 1 und ziehst dann mit dem **TEXTWERKZEUG** ein Textfenster am unteren Bühnenrand auf. Hier gibst du als Text „Gefressen:“ ein.

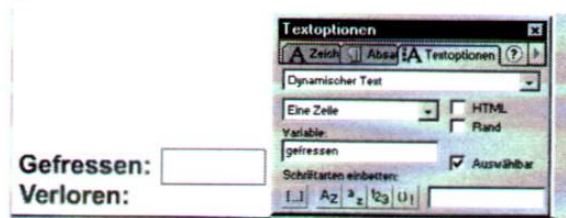
Direkt darunter erstellst du ein weiteres Textfenster und gibst hier den Text „Verloren:“ ein.



Für das Design, also die Auswahl von Schriftart, Farbe etc., gibt es das Menüfenster **ZEICHEN**. Dieses öffnest du, indem du mit der rechten Maustaste auf den zu bearbeitenden Text klickst und dann im Bereich **BEDIENFELDER** auf **ZEICHEN**.

Sind die Beschriftungstexte geschrieben, müssen hinter jeden dieser Texte weitere Textfenster eingefügt werden. Diese Textfenster haben dann allerdings keinen geschriebenen Text, sondern die Zuweisung einer **Variablen**. Hierfür ziehst du dir zunächst ein Textfenster hinter „Gefressen:“.

Im **ZEICHEN**-Menü klickst du nun auf **TEXTOPTIONEN**. Hier ist **STATISCHER TEXT** ausgewählt. Setze diesen Eintrag auf **DYNAMISCHER TEXT**. In den nun erscheinenden Optionen kannst du unter **VARIABLE** den Namen der **Variablen** eingeben, die in diesem Textfeld angezeigt werden soll – also in diesem Fall **gefressen**.



Und schon hast du ein dynamisches Textfeld, das dir den Wert der Variablen **gefressen** anzeigt.

Damit nun auch die Variable **verloren** angezeigt wird, fügst du hinter dem Text „Verloren:“ ein weiteres Textfenster ein und gibst hier unter der Option VARIABLE **verloren** ein.

Hochzählen von Punkten

Zuerst erstellst du nun die Programmierung für den Punkttestand „Gefressen“. In dem ACTIONSCRIPT(), in dem du per **hitTest** die Kollision überprüfst, gibt es die Stelle, an der du den Futterfilm per **removeMovieClip()** aus dem Film löschst.

Genau hier ist der richtige Punkt für das Hochzählen des Punkttestandes. Um das zu erreichen, musst du folgende ACTIONSCRIPT()-Zeile eintragen:

```
_root.gefressen++;
```

Dadurch wird die Variable **gefressen** um einen nach oben gezählt. Trage diese Zeile einfach unter jeder **removeMovieClip()** – Zeile ein:

```
onClipEvent (enterFrame) {
  this._visible = false;
  if (this.hitTest(_root.futter0)) {
    → removeMovieClip(_root.futter0);
    → _root.gefressen++;
    → _level0.Fisch.gotoAndPlay(2);
  }
  if (this.hitTest(_root.futter1)) {
    → removeMovieClip(_root.futter1);
    → _root.gefressen++;
    → _level0.Fisch.gotoAndPlay(2);
  }
  if (this.hitTest(_root.futter2)) {
    → removeMovieClip(_root.futter2);
    → _root.gefressen++;
    → _level0.Fisch.gotoAndPlay(2);
  }
}
```

Nicht vergessen: entsprechend auch unter dem Script für den nach rechts fressenden Fishbud eingeben, sonst zählt er nur hoch, wenn du nach links schwimmend frisst!

Nun fällt folgendes auf: Im Augenblick läuft der Film in einer unendlichen Schleife, die immer wieder bei Frame 1 anfängt, wenn sie den letzten Frame 170 erreicht hat.

Dummerweise steht in Frame 1 der Befehl für die Festlegung der Variablen

gefressen mit dem Wert 0. D.h. selbst nach erfolgreichem Fressen aller drei Portionen wird der Wert von **gefressen** wieder auf 0 gestellt. Das bedeutet, dass Flash nicht wieder von Frame 1 anfangen darf, wenn das Ende des Filmes erreicht ist. Hierfür nutzt du folgenden ACTIONSCRIPT Befehl:

```
gotoAndPlay(2);
```

Dieser Befehl funktioniert genau so wie man ihn übersetzen würde. Er geht an eine bestimmte Stelle und spielt von dort den Film ab. In der Klammer steht die Framezahl als Ziel. Wenn der Film also in Frame 2 weitergehen soll, dann muss in den letzten Frame des Films dieser Befehl als AKTION hinterlegt werden.

Hierfür gehst du einfach in Ebene 2 auf den Frame 170. Mit einem Klick per rechter Maustaste und Auswahl von AKTIONEN gelangst du in den Scriptmodus und gibst hier den ACTIONSCRIPT() Befehl **gotoAndPlay(2);** ein. Dieser sagt: **Gehe zu Frame 2 und Spiele weiter ab**

Damit wäre das Problem mit der Rückstellung der Variablen gelöst. Da du mit dem Klonen der Futterportionen erst in Frame 10 anfängst, fällt der erneute Start in Frame 2 als Zeitverkürzung nicht auf und stört somit nicht den Spielfluss.

Wenn du nun den Film testest, siehst du, wie der Punkttestand bei „Gefressen“ stetig wächst, wenn du etwas gefressen hast.

Beim Hochzählen der Variablen **verloren** muss nun nun die Variablen **verloren** dort erhöht werden, wo der Frame 160 von Futterani geladen wird, d.h. der Frame, in dem das Futter den Boden erreicht. Also klickst du auf Ebene 3 und wechselst per Doppelklick auf das Originalfutter in den Bearbeitungsmodus für *Futterani*.

Hier gehst du in den Frame 160, in dem bereits der ACTIONSCRIPT() Befehl **stop();** steht, und programmierst hier das Hochzählen – und zwar genau wie

beim Punktestand für **gefressen** mit dem Befehl `_root.verloren++;`

Da die Klone eines Symbols immer alle Eigenschaften des Originals übernehmen, enthält nun auch jeder Klon diese Zeile in seinem Frame 160.

Immer dann, wenn ein Klon den Boden erreicht und der Frame 160 angezeigt wird, wird auch die Variable „verloren“ hochgezählt.

```
stop();
_root.verloren++;
```

Problemfall: Beim Durchlauf der ersten Runde fällt das "Original" Futter auch bis zum Frame 160 und zählt damit den Wert für „Verloren“ ebenfalls um eins hoch.

Das Originalfutter fällt faktisch nur ein einziges mal herunter und bleibt dann auf dem Frame 160 stehen. D.h. der Wert für **verloren** wird nur in der ersten Runde falsch gezählt.

Wenn du z.B. die Runden mitzählst, kannst du per Abfrage prüfen, welche Runde gerade läuft. Beginnt die 2. Runde, musst du lediglich den Wert von **verloren** um eins minimieren – und schon hast du dieses „Mitzählen“ der unsichtbaren Original-Portion korrigiert.

Hierfür brauchst du zunächst eine weitere Variable. In Ebene 2 Frame 1 hast du deine Variablenliste. Hier fügst du nun eine neue Variable mit dem Namen „runde“ ein:

```
var richtung = "rechts";
var gefressen=0;
var verloren =0;
var runde=0;
```

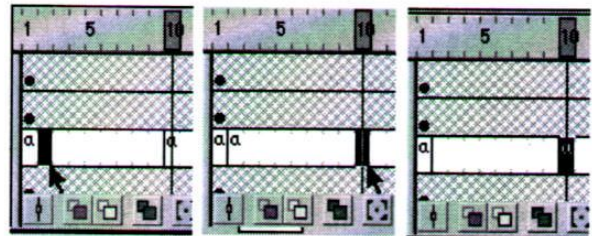
Der Startwert für diese Variable ist 0! Als Ort für das Hochzählen der Variable kommt der Frame 2 in Betracht – hierhin springt Flash nach dem Durchlauf des Filmes.

Da du nach dem Ablauf des Films automatisch in den Frame 2 springst, gehört in diesen Frame auch die nächste Programmierung, also das Hochzählen der Variablen **runde** sowie das Korrigieren der Variablen **verloren**. Da du schon weitere Schlüsselbilder innerhalb der

Zeitleiste für Ebene 2 hast, ist die Platzierung des Scriptes nicht ganz einfach. Versuchst du den Frame 2 anzuklicken, verwandelt sich der Mauszeiger automatisch in eine Hand, und der Framebereich 1-10 wird schwarz markiert.

Um den Frame 2 zu erreichen, klickst du mit der Maus den Frame 10 an – den Frame, wo das Klonen des Futters erfolgt –, hältst die Maustaste fest und ziehst die Markierung auf Frame 2.

Wenn du die Maustaste loslässt, siehst du, dass der Inhalt von Frame 10 nun auf Frame 2 liegt. Nun ziehst du diesen Frame wieder an seinen Platz zurück, also auf Frame 10.



Als Ergebnis hast du danach in Frame 2 ein leeres Schlüsselbild, in dem du die Abfrage der Runde programmieren kannst. Hier die Programmlogik:

```
Wenn die Variable "runde" gleich 1 ist {
vermindere die Variable "verloren" um eins }
zähle die Variable "runde" um eins hoch
```

Und so sieht das Ganze dann in ACTIONSCRIPT() aus:

```
if (_root.runde==1) {
_root.verloren--;
}
_root.runde++;
```

Im Einzelnen:

- `if (_root.runde==1) {Befehle}`
In der Schleife steht die Bedingung die erfüllt sein muss, damit die nachfolgenden Befehle ausgeführt werden. Bei den mathematischen Abfragen größer, kleiner oder gleich musst du folgende Schreibweisen einhalten:
`<=` ist kleiner oder gleich

\geq ist größer oder gleich

$=$ ist gleich

In diesem Fall wird gefragt, ist `_root.runde` gleich 1

- `_root.verloren--;`
Diese Zeile vermindert den Wert der Variablen `verloren` um eins. `++` würde den Wert um eins erhöhen, wie in der nächsten Zeile:
- `_root.runde++;`
Diese Zeile erhöht den Wert der Variablen `runde` um eins.

Wird das Spiel gestartet und der Frame 2 zum ersten Mal durchlaufen, steht der Wert der Variablen `runde` auf 0. Das bedeutet, dass nicht der Inhalt der geschweiften Klammer ausgeführt wird, sondern lediglich das Hochzählen der Variable `runde` um 1.

Läuft dann die zweite Runde, steht der Wert der Runde "noch" auf 1, und damit wird der Inhalt der geschweiften Klammer ausgeführt. Da danach der Wert von der Variablen `runde` immer wieder um eins nach oben gezählt wird, verhinderst du, dass der Inhalt der geschweiften Klammer ein weiteres Mal durchlaufen wird.

Nun zählt Flash sowohl die gefressenen Portionen als auch die verlorenen Portionen – und zwar richtig!

Hintergrundgrafik und Spieldesign

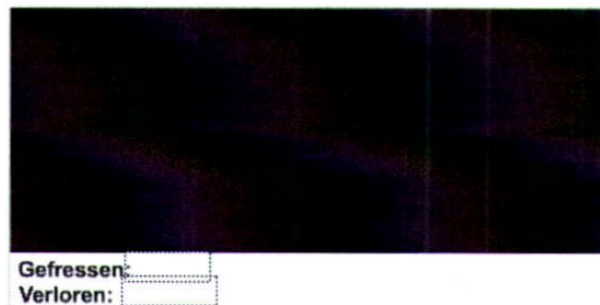
Bevor ich nun zur Punkteprüfung und zur „Game Over“-Seite komme, fehlt noch eine wichtige Sache: das Grunddesign des Spielfeldes. Fishbud schwimmt laut Gameplay in einem Aquarium.

Dieses Aquarium wäre nun einzufügen. Als Ebene hierfür schlage ich die Ebene 4 vor, in der du bisher die Punktestände anzeigst. Für das Aquarium selbst kannst du das RECHTECK-WERKZEUG benutzen. Damit kannst du ein blaues Rechteck mit farbigem Rand als Aquarium zeichnen:

Du klickst zunächst in Szene 1 auf Ebene 4. Dann wählst du in der Werkzeugleiste das RECHTECKWERKZEUG aus und ziehst über die Bühne ein Rechteck

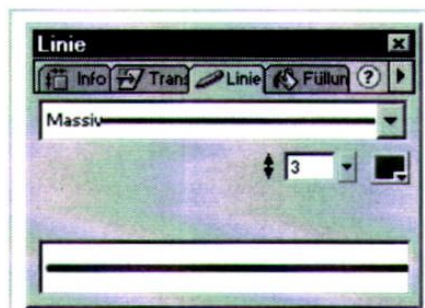
von einer der oberen Ecken bis knapp über den Punktestandtext.

Anschließend markierst du das Rechteck mit dem UNTERAUSWAHL-WERKZEUG und gruppierst es per MODIFIZIEREN|GRUPPIEREN.



Da Flash den Rahmen und die Füllung eines geometrischen Objektes immer getrennt voneinander einfügt, erleichtert das Gruppieren die Arbeit erheblich.

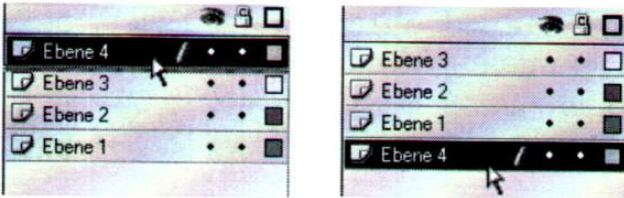
Wünschst du um die blaue Wasserfläche einen dickeren Rahmen oder/und eine andere Farbe, wählst du die Linienstärke und -farbe im Registerblatt LINIE des Menüfenster INFO aus.



Die Farben für Rahmen und Füllung kannst du übrigens auch über das Werkzeugmenü auswählen.

Nun fällt auf, dass dieses Rechteck alles andere unter sich versteckt. Jede „höhere“ Ebene liegt auf der unteren Ebene. Im Augenblick arbeitest du in Ebene 4 und liegst damit über allen anderen Ebenen.

Doch zum Glück ist Flash in diesen Dingen nicht absolut starr: Du kannst mit der Maus per Drag and Drop die Ebene 4 mit der linken Maustaste anklicken und per gedrückter Maustaste unter die Ebene 1 ziehen. Im Resultat legt sich das Rechteck hinter alles andere.

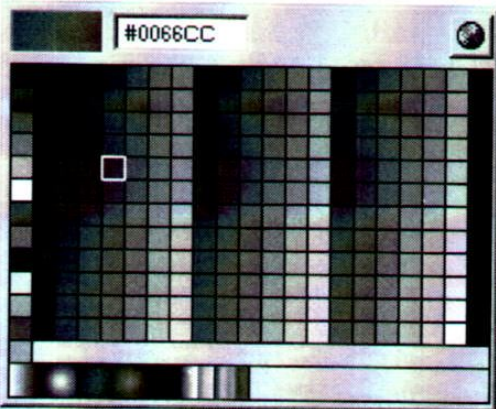


Nun siehst du, dass Fishbud von weißer Farbe umgeben ist. Beim Bitmap-Nachzeichnen hat Flash die freien Flächen automatisch mit der Hintergrundfarbe gefüllt. Zu diesem Zeitpunkt gab es noch keinen Hintergrund, weswegen diese Bereiche weiß sind. Willst du diese Umrandungen loswerden, hast du zwei Möglichkeiten.

- 1.) Du löschst sie einfach heraus. Hierfür klickst du doppelt auf Fishbud, so dass du in den Bearbeitungsmodus gelangst. Hier klickst du Fishbud einfach an und wählst **MODIFIZIEREN|GRUPPIERUNG AUFHEBEN**. Danach kannst du die weißen Flächen doppelt anklicken und sie über **BEARBEITEN|LÖSCHEN** aus Fishbud entfernen. Fertig.

Vergiss nicht, dass du die weißen Flächen in allen vier Erscheinungsformen löschen musst, also auch in den Frames 2, 3 und 4.

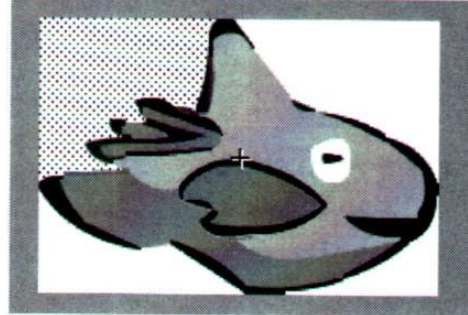
2. Du musst die einzelnen Bilder von Fishbud manuell mit der blauen Farbe füllen, die das Wasser in deinem Becken hat. Hierfür merkst du dir die Farbe, die du für das Wasser ausgewählt hast.



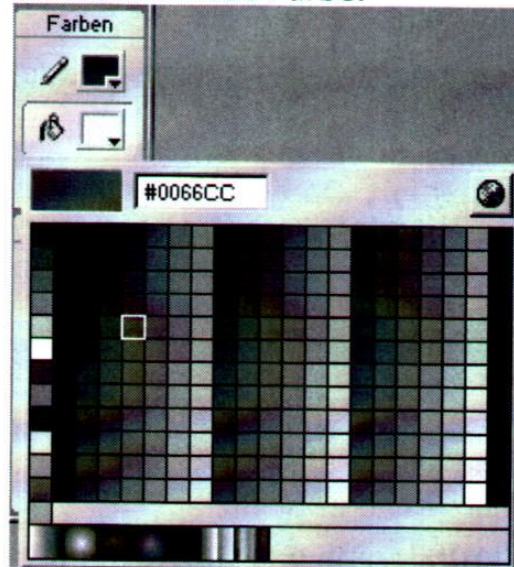
Dann klickst du doppelt auf Fishbud, damit du in den Bearbeitungsmodus gelangst.

Für die Bearbeitung vergrößerst du am besten die Ansicht über **ANSICHT|VERGRÖßERN**

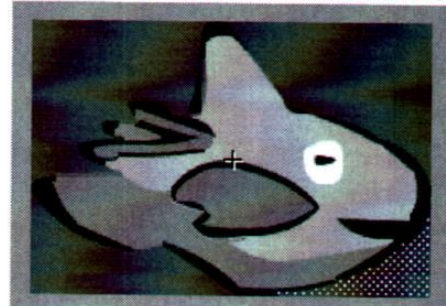
Hier klickst du auf Fishbud, und dann doppelst du auf die erste weiße Fläche, so dass diese markiert erscheint.



Danach wählst du beim Fülleimer die Farbe deines Wassers aus. Die ausgewählte Fläche erhält dann sofort ihre neue Farbe.

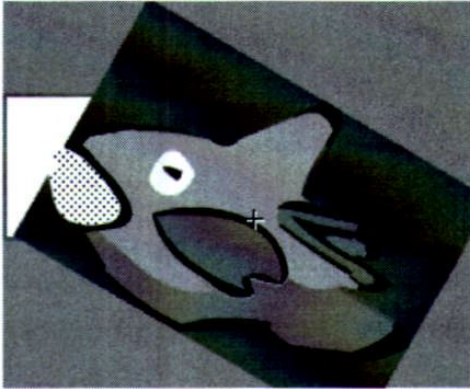


Nun füllst du so, nach und nach, jede weiße Fläche um Fishbud herum.



Da der Hintergrund inaktiv ist, erscheint er in diesem Bearbeitungsmodus heller. Die Bildschirmansicht macht also den Eindruck, als füllst du mit der falschen Farbe – aber keine Sorge, es hat alles seine Richtigkeit!

Vergiss nicht, in allen vier "Erscheinungsformen" von Fishbud die weißen Flächen zu füllen!



Nun fehlt noch ein wenig Flora – was wäre schließlich ein Aquarium ohne Pflanzen?! Hierfür habe ich im Internet eine „Wasserlandschaft“ als Grafik gefunden. Diese Datei heißt **WASSERLANDSCHAFT.GIF** und liegt auf www.spieleinflash.de zum Download bereit. Über DATEI|IMPORTIEREN gelangst du in die Dateiauswahl.

Über MODIFIZIEREN|BITMAP NACHZEICHNEN solltest du dieses **GIF** in eine Vektorgrafik umwandeln!

Diese Grafik positionierst du am besten am unteren Aquariumrand. Damit die Grafik auch genau hinein passt, kannst du sie per MODIFIZIEREN|SKALIEREN in ihrer Größe anpassen.



Nun sieht das ganze schon wie ein echtes PC-Game aus. Am unteren Rand werden die Punktestände angezeigt, und du hast eine passende Spielegrafik. Diesen Stand kannst du dir auf www.spieleinflash.de herunterladen. Die Datei heißt **FISHBUD5.FLA**.

Spielstandabfrage – Neues Level oder Ende des Spiels

Nun kommt der Schlüsselteil des 3. Aktes. In dem Moment, wo der Wert der Variable **verloren** größer ist als 5, soll das Spiel zu Ende sein – d.h. also

(**_root.verloren**>5). Als Ort für das passende ACTIONSCRIPT() bietet sich die Stelle an, an der du die Variable **verloren** hoch zählst, also im Frame 160 der Animation des Futters.

In diesem Frame liegt im Augenblick das ACTIONSCRIPT() für das Stoppen dieser Filmsequenz **stop()**; und das Hochzählen der Variablen **verloren**. Hier muss nun die Abfrage hinzu, dass der Spielablauf unterbrochen werden soll, wenn der Wert der Variablen **verloren** größer ist als 5.

Denke daran, dass Flash wie ein Dauerkino funktioniert. Damit das Spiel unterbrochen wird, muss die „Game Over“-Seite an einer Frameposition liegen, in der die Tetrisbalken des normalen Spiels nicht vorkommen. Als Vorschlag gebe ich den Frame 250 im Hauptfilm vor.

Zunächst widmen wir uns aber der Scripterweiterung. Du klickst das Futter doppelt an, so dass sich der Bearbeitungsmodus für die Filmsequenz **futterani** öffnet. Hier scrollst du die Zeitleiste bis zum Frame 160 vor, klickst mit der rechten Maustaste auf diesen Frame und wählst im Menü AKTION.

Am besten schaltest du den Expertenmodus ein!

Unter die vorhandenen Zeilen gehört nun folgende Programmlogik:

Wenn (die Variable **verloren größer ist als 5) dann**

{springe im Hauptfilm zum Frame 250 und bleibe dort stehen}

In ACTIONSCRIPT() sieht das so aus:

```
stop ();
_root.verloren++;
if (_root.verloren>5) {
_level0.gotoAndStop(250);
}
```

Bevor du dies nun testen kannst, musst du im Hauptfilm auf den Frame 250 gehen und dort einen „Inhalt“ einfügen. Also klickst du auf Szene 1 und scrollst die Zeitleiste bis zum Frame 250. Im Augenblick hören alle vorhandenen

Tetrisbalken bei Frame 170 auf, und im Frame 170 steht dein Befehl, dass Flash zum Frame 2 springen soll. Der Frame 250 wird also im normalen Spielablauf niemals erreicht, es sei denn der Wert der Variablen `verloren` ist größer als 5. In diesem Fall schickt ihn dieses `ACTIONSCRIPT()` hierhin.

Im Augenblick ist es egal, welche Ebene du für den "Game Over"-Text nimmst; ich schlage die Ebene 1 vor. Willst du nun einen Inhalt im Frame 250 einsetzen, musst du ein weiteres Schlüsselbild einfügen.

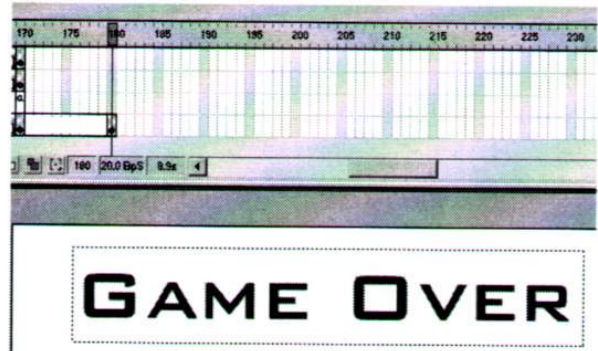
Hier ist nur dumm, dass Flash sowohl beim Einfügen eines leeren als auch bei dem eines normalen Schlüsselbildes immer den vorangegangenen „Tetrisbalken“ verlängert.

Willst du verhindern, dass beim Erstellen eines neuen Schlüsselbildes in Frame 250 automatisch der „Tetrisbalken“ von Fishbud verlängert wird, musst du zunächst einmal Flash klar machen, dass der vorangegangene Balken am Ende seiner Anzeigezeit angelangt ist. Also klickst du mit der rechten Maustaste auf den Frame 171 und wählst im Menü **LEERES SCHLÜSELBILD EINFÜGEN**.

Das leere Schlüsselbild informiert Flash darüber, dass die vorangegangene Filmsequenz nicht mehr fortgesetzt wird.

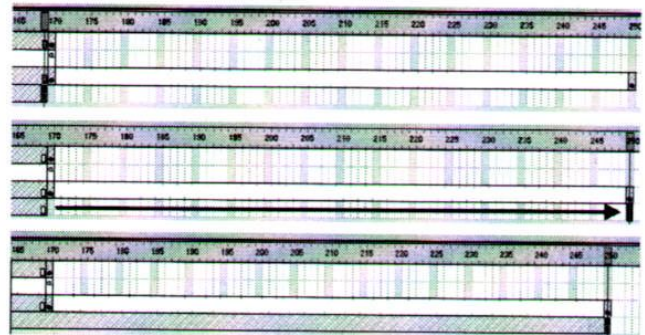
Hättest du z.B. bei Frame 250 ein leeres Schlüsselbild eingesetzt, würde Flash annehmen, dass der „Tetrisbalken“ von Fishbud bis zum Frame 249 liefe, und verlängerte ihn automatisch bis dort!

Anschließend klickst du mit der rechten Maustaste auf den Frame 250 und wählst **SCHLÜSELBILD EINFÜGEN**. Nun kannst du z.B. mit dem **TEXTWERKZEUG** den Text Game Over einbauen.



Damit der Spieler weiß, wieviel Portionen er gefressen hat, sollte auch in diesem Fenster die Ebene 4 sichtbar sein. Hierfür kannst du die Ebene 4 einfach bis zum Frame 250 verlängern.

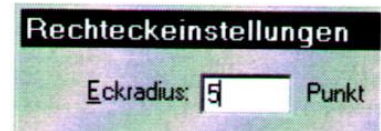
Optisch bleiben dann das Aquarium und die Punktanzeige sichtbar. Hierfür löschst du das Schlüsselbild in Frame 170 in Ebene 4 und verlängerst den Balken, indem du mit gedrückter Maustaste den Endframe bis zum Frame 250 ziehst.



Hat das Spiel diese Stelle erreicht, sollte dem Spieler die Option angeboten werden, noch einmal zu spielen oder aber an einem Gewinnspiel teilzunehmen.

Für die Auswahl dieser beiden Optionen setzt du hier am besten zwei Schaltflächen ein. Diese Buttons kannst du selber erstellen, z.B. mit dem **RECHTECKWERKZEUG**.

Für eine schönere Optik kannst du in der Werkzeug-



leiste unter **OPTIONEN** den Wert für den Eckradius eingeben – so erhält der Button „runde“ Ecken.

Außerdem kannst du unter FARBEN die Farben für den Rahmen und die Füllung auswählen.

Möchtest du die Farben für das Rechteck nachträglich ändern, wählst du den Bereich, den du ändern möchtest, per Mausklick aus.



Möchtest du den Rahmen ändern, musst du ihn doppelt anklicken! Wähle dann bei ZEICHENFARBE eine Farbe. Die Farbe des Rahmens ändert sich on the fly.

Beim Ändern der Füllung klickst du einfach auf die Füllung deines Rechtecks und wählst eine FÜLLFARBE.

Ist der erste Button gezeichnet, markierst du ihn per Doppelklick auf den Füllbereich und gruppierst ihn über MODIFIZIEREN|GRUPPIEREN. Bevor du nun die Beschriftung machst, solltest du dir eine Kopie von diesem Button für die zweite Schaltfläche erstellen.

Du klickst den Button einfach an, wählst BEARBEITEN|KOPIEREN und dann BEARBEITEN|EINFÜGEN. Diesen neuen Button kannst du dann mit der Maus an die richtige Stelle schieben. Nun kannst du mit dem TEXTWERKZEUG die Beschriftungen für deine Knöpfe machen, z.B. „Gleich noch mal spielen“ und „Ab zum Gewinnspiel“.



Schreibst du, wie in diesem Fall, zweizeilig, überlasse den Zeilenumbruch nicht Flash, sondern setze ihn selber mit der RETURN-TASTE ein – sonst ist der Text später vielleicht nicht ganz zu lesen!

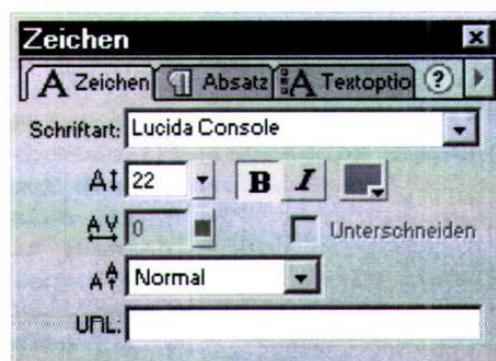
Am besten gruppierst du die Buttons mit ihren Texten. Hierzu nimmst du das UNTERAUSWAHLWERKZEUG und hältst die UMSCH-TASTE gedrückt. Dann klickst du den Button und seinen Text an und wählst MODIFIZIEREN|GRUPPIEREN.

Anschließend ist die Umwandlung in ein Symbol an der Reihe: du klickst einen der Buttons an und wählst EINFÜGEN|IN SYMBOL KONVERTIEREN. Als NAME vergibst du **Button1** und als VERHALTEN wählst du SCHALTFLÄCHE.

Entsprechend verfahrst du mit der anderen Schaltfläche, die du **Button2** nennst. Klickst du nun eine der beiden Schaltflächen doppelt an, gelangst du in den Bearbeitungsmodus für diese Schaltfläche.



Im oberen Bereich siehst du vier Frames mit den Titeln AUF, DARÜBER, DRÜCKEN, AKTIV. Soll sich später im Spiel der Button ändern, wenn er mit der Maus berührt wird, musst du im Frame DRÜBER eine geänderte Version dieses Buttons einfügen. Soll sich z.B. sich die Textfarbe ändern, klickst du zunächst mit der rechten Maustaste auf den DRÜBER-FRAME und wählst SCHLÜSSELBILD EINFÜGEN. Dann klickst du den Text doppelt an und wählst über das ZEICHEN-Menü eine andere Farbe.



Gelangt der Spieler nun auf die „Game Over“-Seite und fährt mit der Maus über diesen Button, ändert sich die Farbe des Textes. Hier kannst du auch weitere Änderungen einbringen, z.B. eine Farbänderung des Buttons.

Die Anzeigearten DRÜCKEN und AKTIV kannst du entsprechend bearbeiten und bestimmen. Die Anzeige von DRÜCKEN erfolgt, wenn der Knopf gedrückt wird; die Anzeige von AKTIV dann, wenn der Button gedrückt wird und weiterhin sichtbar ist. Da in diesem Fall die Knöpfe nach dem Klick nicht mehr sichtbar sein werden, genügt die Änderung unter DRÜBER. Hast du deine Änderungen vorgenommen, dann bearbeitest du entsprechend den zweiten Button

Nun kommen wir zu den ACTIONSCRIPT()-Aktionen, die erfolgen sollen, wenn der Spieler einen der beiden Knöpfe angeklickt hat. Beim Button1 soll das Spiel wieder von vorne beginnen; bei Button2 soll sich eine Internetseite mit einem Eingabeformular für die Teilnahme an einem Gewinnspiel öffnen.

Button1 – Gleich noch einmal spielen

Zunächst kehrst du per Klick auf Szene 1 in den Bearbeitungsmodus für den Hauptfilm zurück. Hier scrollst du mit der Maus zu Frame 250 vor und klickst in Ebene 4 auf den Frame 250. Dann klickst du mit der rechten Maustaste den Button1 an und wählst im Menü den Punkt AKTIONEN.

Expertenmodus nicht vergessen!
STRG+E oder Klick auf die Pfeiltaste direkt unter der SCHLIEBSCHALTFLÄCHE des Dialogfeldes.

Nun musst du Flash mitteilen, dass es bei Mausklick auf diesen Button wieder bei Frame 1 beginnen soll. Den Sprung zu Frame 1 und das erneute Abspielen erreichst du mit dem nun schon bekannten Befehl `gotoAndPlay()`.

Zur Abfrage des Mausklicks dient der Befehl `on(release)`. Wörtlich übersetzt heißt dies „beim loslassen“. Nun – wenn man die Maustaste loslässt, bedeutet dies sinngemäß ein Klick. Das erfordert folgende Programmlogik:

```
Beim Maustaste loslassen{ gehe zum Frame1 und spiele ab}
```

In Flash sieht dies dann so aus:

```
on (release) {  
gotoAndPlay(1);  
}
```

Damit ist der erste Knopf programmiert.

Button 2 – Ab zum Gewinnspiel

In diesem Fall soll eine Internetseite aufgerufen werden, der der Punktestand der gefressenen Portionen mitgeteilt wird. Der Spieler kann dann seine Angaben machen und per E-Mail an einem Gewinnspiel teilnehmen.

Hierfür schlage ich eine Kombination von HTML und PHP vor. PHP gehört heute zu den Standards und steht nahezu auf jedem Server zur Verfügung. Diese Scriptsprache bindest du direkt in deine HTML-Dateien ein, ähnlich wie JAVASCRIPT().

Mit Flash rufst du also eine „normale“ Internetadresse basierend auf HTML UND PHP auf und verarbeitest hier den Punktestand des Spieles. Der Aufruf einer Internetseite erfolgt in ACTIONSCRIPT() mit dem Befehl GETURL(). Die aufzurufende Internetseite heißt **TEILNAHME.PHP**.

Durch die Verwendung von PHP-Script in einer HTML-Datei verändert sich der Dateityp von HTML in PHP. So weiß der Server, dass er den PHP-Interpreter aktivieren muss, um diese Internetseite richtig darstellen zu können. **TEILNAHME.PHP** ist also eine normale HTML-Datei mit ein paar Zeilen PHP-Script.

Der Vorteil von PHP ist, dass du beim Aufruf der URL auch gleich die Variablen übergeben kannst. Die Schreibweise sieht so aus:

TEILNAHME.PHP?ANZAHL=_ROOT.GEFRESSEN

Die Programmlogik sagt:

Beim Maustaste loslassen {rufe die Datei `teilnahme.php` auf und Übergabe den Wert von `_root.gefressen` als Variable mit dem Namen `anzahl`}

Das `ACTIONSCRIPT()` sieht so aus:

```
on (release) {
    getURL
    ("teilnahme.php?anzahl="+_root.gefressen);
}
```

Wichtig ist hier die Schreibweise. In den Hochkommata kommt der **DATEINAME**, mit dem **+ZEICHEN** stellst du eine logische **UND-VERKNÜPFUNG** her und fügst dann durch die Nennung des Variablen Namens den Wert der Variable ein.

`GetURL("Dateiname"+Variable);`

Als nächstes musst du die HTML/PHP Datei **TEILNAHME.PHP** aufbauen. Hier muss lediglich ein Standardformular in HTML generiert werden. Hier ein einfaches Formular als Basis:

```
<html>
<head>
<title>Gewinnspiel Teilnahme
FishBud</title>
</head>
<body bgcolor=#ffffff>
<font face=arial size=2>
<form>
Name: <input type="Text" name="Name"
size="30"><br>
Straße: <input type="Text"
name="Strasse" size="30"><br>
Ort: <input type="Text" name="Ort"
size="20"><br>
E-Mail: <input type="Text" name="Mail"
size="30"><br>
<input type="Submit" name="Senden"
value="">
</form>
</font>
</body>
</html>
```

Für PHP-Scripte gibt es einen ähnlichen Tag wie für `JAVASCRIPT()`. Dieser heißt

`<?php ...Befehle ...?>`

Damit wir hier auch ein wenig PHP nutzen, können wir per **Wenn-Dann-Sonst-Abfragen** einen Glückwunschtext einbauen, der die Anzahl der gefressenen Portionen bewertet.

- Bei über 50 Portionen soll folgender Text erscheinen:
"Bombastisch, du hast xxx Portionen gefressen. Damit hast du eine Riesenchance auf den HighScore Preis."
- Bei über 30 Portionen erscheint:
"Sensationell, du hast xxx Portionen gefressen."
- Bei unter 30 Portionen sieht man:
"Super, du hast xxx Portionen gefressen."

Genau wie im `ACTIONSCRIPT()` nutzt du hierfür die **IF-Abfrage**. Hier ist der Code für die erste Abfrage:

```
if ($anzahl>=50)
{echo"Bombastisch - Du hast
$anzahl Portionen gefressen!
<br>Damit hast du eine
Riesenchance auf den Highscore-
Preis!";}
```

Eine Besonderheit in PHP ist das führende `$`-Zeichen vor Variablen. Dadurch unterscheidet PHP eindeutig zwischen Text und Variablenwerten und kann problemlos beides kombinieren. In diesem Fall erscheint an Stelle des Textes `$anzahl` der Wert dieser Variablen.

Mit `echo "...text...";` generierst du an dieser Stelle im HTML-Dokument eine Textausgabe. Schriftformatierungen, die du vorher in HTML eingebaut hast, haben auch für diesen erstellten Text ihre Gültigkeit. Das geänderte Formular sieht so aus:

```
<html>
<head>
<title>Gewinnspiel Teilnahme
FishBud</title>
</head>
<body bgcolor=#ffffff>
<font face=arial size=2>
<?php
if ($anzahl>=50)
{echo"Bombastisch - Du hast $anzahl
Portionen gefressen!
<br>Damit hast du eine riesen Chance
auf den Highscore-Preis!";}
else {
if ($anzahl>=30)
{echo"Sensationell - Du hast $anzahl
Portionen gefressen!";}
else {
```

```

echo"Super - Du hast $anzahl Portionen
gefressen!";}
}
}
?>
<form>
Name: <input type="Text" name="Name"
size="30"><br>
Straße: <input type="Text"
name="Strasse" size="30"><br>
Ort: <input type="Text" name="Ort"
size="20"><br>
E-Mail: <input type="Text" name="Mail"
size="30"><br>
<input type="Submit" name="Senden"
value="">
</form>
</font>
</body>
</html>

```

Fehlt noch ein Formularfeld, in dem du den Wert von `$anzahl` hinterlegst – schließlich muss dieser Wert ebenfalls übertragen werden. Als PHP `echo "..."`; Befehl generierst du nun ein Formularfeld des Typs `hidden`; der `value` soll gleich `$anzahl` sein. So sieht das PHP-Script aus:

```

<?php
echo"<input type=hidden
name='punkte' value='$anzahl'>";
?>

```

Als Ergebnis erhältst du eine E-Mail mit den Formulareingaben des Spielers sowie der Anzahl der von ihm gefressenen Portionen.

Testest du einen Flashfilm online, vergiss nicht, dass du den Film mit der Maus anklicken musst, damit z.B. die Steuerung funktioniert!

Ich habe hier keine FORM ACTION hinterlegt. Dieses Formular ist so nicht verwendbar! Bitte erweitere dieses Formular so, dass es z.B. per CGI-Script "Formsend" oder ähnlich gesendet wird. Dein Provider gibt dir alle notwendigen Daten.

Bitte beachte, dass PHP eine serverseitige Scriptsprache ist. Wenn du also auf deinem PC keine Serversoftware sowie einen PHP-Interpreter installiert hast, kannst du dieses Script nur online testen.

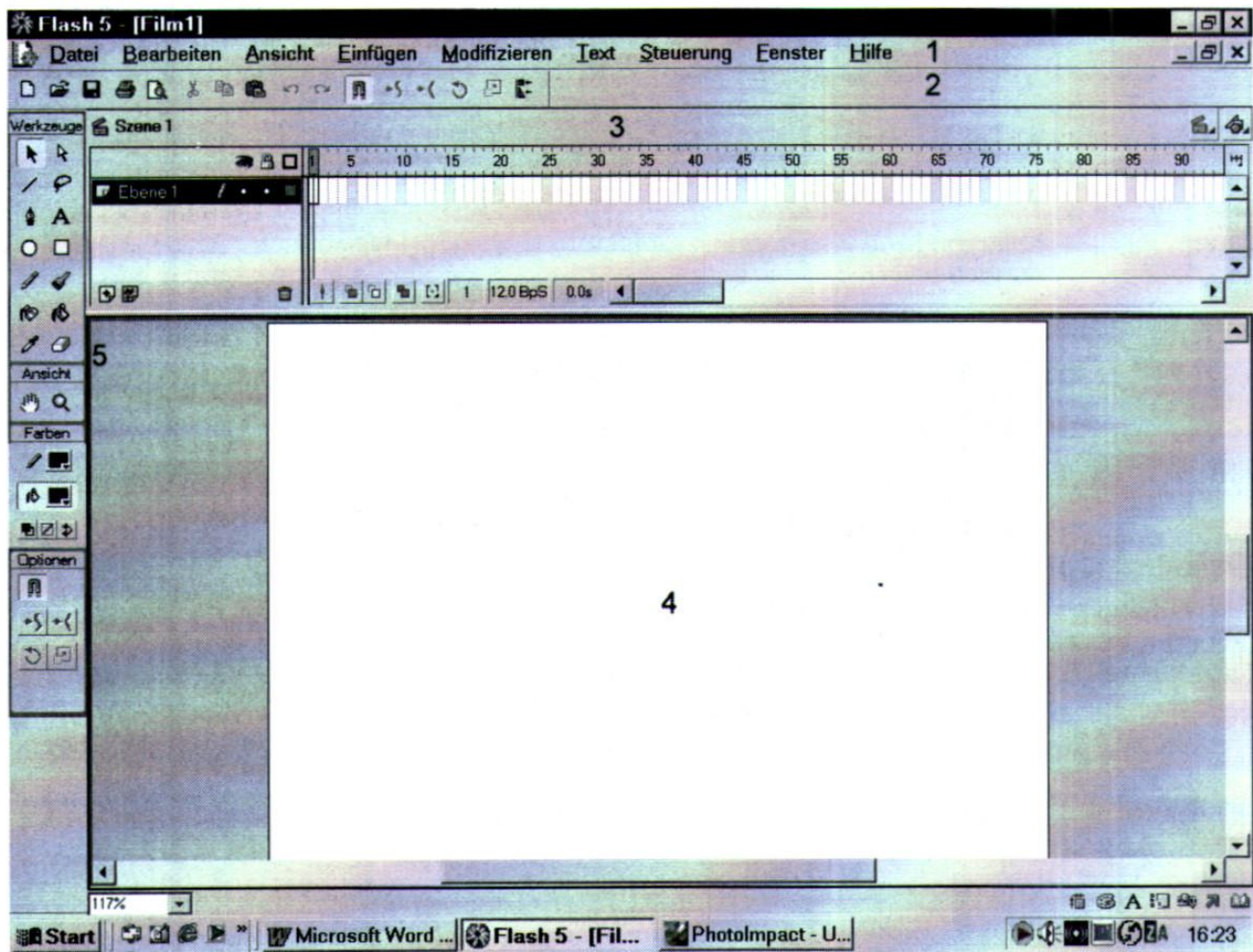
Nun ist das Spiel in seinen Grundfunktionen fertig. Im 3. Akt des Drehbuches musst du nun noch ergänzen, dass es weitere Variablen gibt: Eine für das Zählen der verlorenen Portionen, eine für das Zählen der gefressenen Portionen und eine für das Zählen der Spielrunden.

Ebenso gehört hierhin, dass es eine PHP- Datei gibt, die die Anzahl der gefressenen Portionen in ein HTML-Formular aufnimmt und versandfertig zur Verfügung stellt.

Diesen Stand kannst du als **FISHBUD6.FLA** auf www.spieleinflash.de herunterladen.

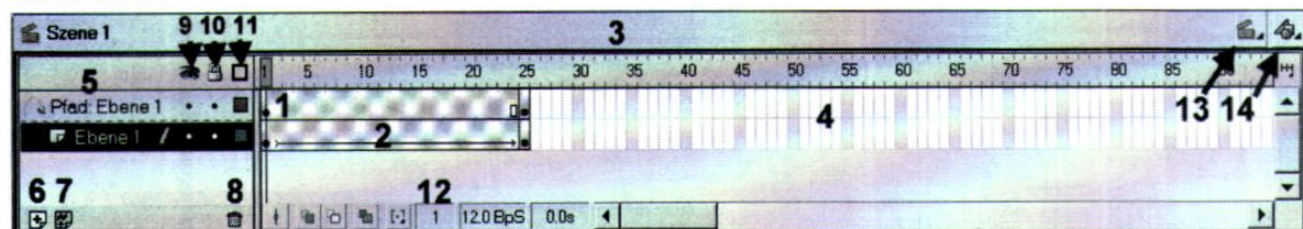
Kurzübersicht Flash 5 – Spielprogrammierung

Arbeitsoberfläche



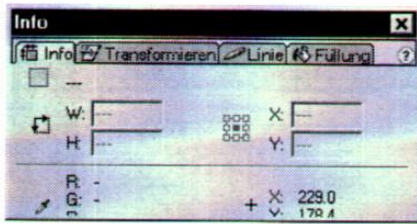
- | | | |
|--------------------|---------------|-------------------|
| 1) Hauptmenüleiste | 3) Zeitleiste | 5) Werkzeugleiste |
| 2) Symbolleiste | 4) Bühne | |

Zeitleiste

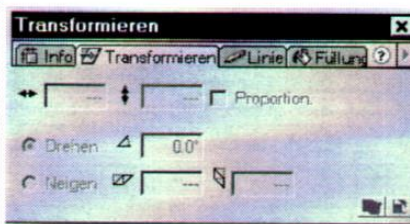


- | | | |
|---------------------|--------------------------------|------------------------|
| 1) Schlüsselbild | 7) Pfadebene einfügen | 13) Szene bearbeiten |
| 2) Bewegungs-Tween | 8) Ebene löschen | 14) Symbole bearbeiten |
| 3) Frameleiste | 9) Ebenen ein-/ ausblenden | |
| 4) einzelne Frames | 10) Ebene sperren / entsperren | |
| 5) Ebenen | 11) Ebene als Kontur anzeigen | |
| 6) Ebene hinzufügen | 12) Nr. des aktiven Frames | |

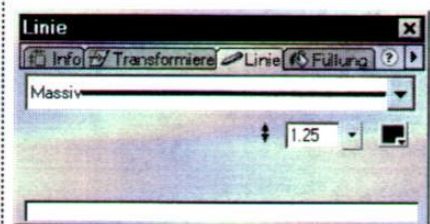
Bedienfelder Einblenden über Fenster| Bedienfelder



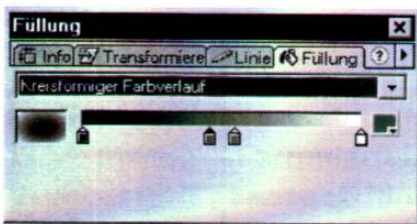
Gibt allgemeine Informationen über die Positionierung und Größe von Symbolen



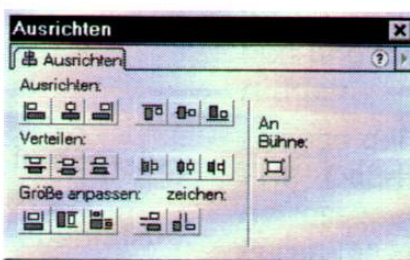
Zum Verformen, Drehen und Neigen von Objekten



Design der Linienart bei gezeichneten Objekten, z.B. bei der Umrandung von Rechtecken, Kreisen etc.



Zum Füllen von gezeichneten Objekten mit verschiedenen Füll-effekten



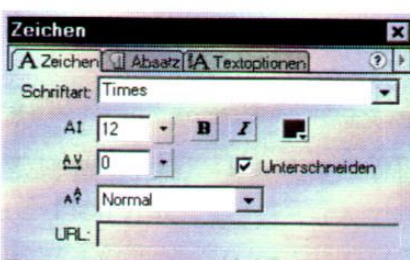
Zum Ausrichten von Objekten, entweder in Relation zueinander oder in Relation zur Bühne



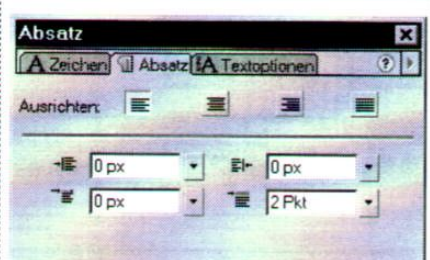
Zur Auswahl von Mischfarben für die Farboptionen der Zeichen- sowie der Füllfarbe



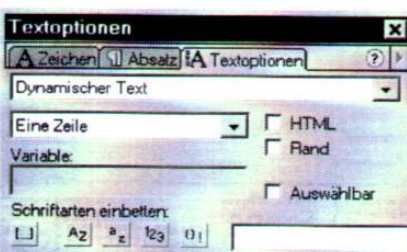
Zur Auswahl von Standardfarben für die Farboptionen der Zeichen- sowie der Füllfarbe



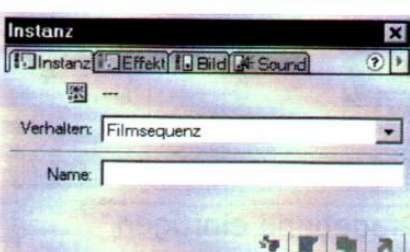
Zur Einstellung von Schriftart, -farbe, -größe sowie Buchstabenabständen



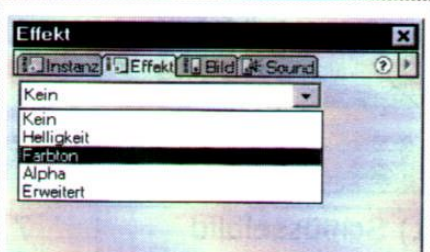
Zur Ausrichtung des Textes



Zur Einstellung eines Textfensters als Statischer oder Dynamischer Text – Zuweisung von Variablen Inhalten und Einstellung für die spätere Veröffentlichung



Verwaltung von Symbolen. Einstellung des Verhaltens, sowie Vergabe des Instanznamens



Zuweisung von Effekten für Symbole.

Bedienfelder einblenden über Fenster| Bedienfelder

<p>Zur Vergabe von Bildnamen bzw. Framenamen. Ebenso kann man hier Tweeningangaben zum Bild/Frame machen</p>	<p>Angaben zum eingebundenem Sound. Wichtig ist hier die Angabe des Wiederholungswertes für lang andauernde Sounds</p>	<p>Zur Verwaltung der einzelnen Szenen. Hier sind Hinzufügen, Löschen und Umbenennen möglich.</p>

Bildaktionen – ActionScript()-Fenster einblenden über Fenster| Aktionen

- | | |
|--|--|
| <p>1) Scriptfenster zur Eingabe</p> <p>2) Optionsmenü z.B. zur Auswahl des Expertenmodus</p> <p>3) Shortcutleiste für die Schnellauswahl von ActionScript()-Befehlen</p> | <p>4) Dropdown-Menü für die strukturierte Auswahl von ActionScript()-Befehlen</p> <p>5) Film – Explorer Sitemap des Flash-filmes inkl. aller Objekte und ActionScripts()</p> |
|--|--|

Markierst du in der Zeitleiste einen Frame, also ein Bild, und wählst FENSTER| AKTIONEN, erscheint dieses Bildaktionen – Fenster. Du kannst auch Objekte mit eigenem ActionScript() versehen. Hierzu wählst du das Objekt aus und wählst dann FENSTER| AKTIONEN.

Nützliche Funktionstasten

F1 – Hilfe

F5 – Bild oder Frame einfügen

F6 – Schlüsselbild einfügen

F7 – leeres Schlüsselbild einfügen

F8 – in Symbol konvertieren

Bewegungstween (durch lila Farbbalken mit schwarzem Pfeil markiert)



- Ein Bewegungstween funktioniert nur mit Symbolen.
- Es beginnt und endet immer mit einem Schlüsselbild. Die Unterschiede dieser zwei Bilder verarbeitet Flash automatisch zu einem Tween.
- Jedes Bewegungstween braucht seine eigene Ebene
- Soll sich die Bewegung eines Objektes nicht geradlinig verhalten, muss eine Pfadebene hinzugefügt werden. In dieser Pfadebene wird der gewünschte Weg des Objektes eingezeichnet und das Objekt dann auf diesem "Pfad" verankert

Testen von Filmen

Grundsätzlich genügt für den Testlauf das Drücken der RETURN-TASTE. Leider funktionieren hierbei oft nicht alle Animationen und Interaktionen. Aus diesem Grund bietet Flash eine dem späteren Fertigfilm sehr nahe kommende Vorabansicht. Diese erreichst du so:

- Du drückst die Tastenkombination STRG-+ ENTER
- Du wählst im Menü STEUERUNG| FILM TESTEN
- Du wählst im Menü STEUERUNG| SZENE TESTEN

Film veröffentlichen

Für das Web muss aus dem **FLA**-Dateityp eine **SWF**-Datei generiert werden. Neben dieser **SWF**-Datei benötigt man dann noch eine **HTML** Datei, die die zum Anzeigen der **SWF**-Datei nötigen TAGS beinhaltet. Unter DATEI| EINSTELLUNGEN FÜR VERÖFFENTLICHUNGEN kannst du für den Export des Flashfilmes weitere Einstellungen vornehmen. Hierzu gehören z.B. die Exportqualität der Grafiken. Neben dem Flashformat kannst du hier auch das Quicktime- oder RealPlayerformat generieren. Sind die Einstellungen erfolgt, kannst du per DATEI|VERÖFFENTLICHEN den Flashfilm sowie die notwendige HTML-Datei erstellen.

Highscoreliste

Neben der Teilnahme an einem Gewinnspiel gibt es in Flashspielen häufig auch eine Highscoreliste. In dieser Liste kann man sich eintragen; ist der erreichte Punktestand gut genug, erhält man dann einen Eintrag in den oberen Reihen.

Sollen die Einträge langfristig gespeichert werden, musst du entweder eine Datenbank nutzen, in der die Einträge gespeichert werden, oder du legst eine Textdatei an, in der jeder einzelne Eintrag als Zeile eingefügt wird.

Dieser Text wird dann einfach ausgelesen, sortiert und angezeigt. Ohne eine solche Speicherung der Einträge hättest du das Problem, dass die Spieler immer nur ihre eigenen Einträge sähen und nicht die der anderen Spieler.

Für die Erstellung einer solchen Liste bietet sich wieder PHP als Scriptsprache an; hier kannst du eine Textdatei erstellen und mit Inhalten füllen. Diese Inhalte kannst du wiederum auslesen und das Ergebnis an Flash zurück geben.

Da dies kein PHP-, sondern ein Flash-Text ist, greife ich auf zwei Scripte von **Sebastian Wichmann** von der Website www.flashhilfe.de zurück. An dieser Stelle danke ich ihm noch mal dafür, dass ich diese Scripte hier verwenden darf.

Die notwendigen Scripte heißen, `laden.php4` und `speicher.php4`. Diese kannst du dir auf www.flashhilfe.de oder auf www.spieleinflash.de herunterladen. Die Datei `LADEN.PHP4`, öffnet eine bestimmte Datei – hier `HIGHSCORE.TXT` –, lädt alle Zeilen dieser Datei ein und sortiert diese nach den erreichten Punktzahlen.

Da eine Highscorelist nicht mit dem kleinsten sondern mit dem größten Wert beginnt, wird die sortierte Liste umgekehrt und schließlich an Flash zurück gegeben. In Flash selbst gibt es dann zwei Textfelder.

Beide Felder sind als dynamischer Text angelegt; eines bezieht sich auf die Variable `namen` und eines auf die Variable `hits`. Diese zwei Variablen werden von dem PHP-Script mit Inhalt gefüllt und an Flash zurück gegeben.

Die Datei `SPEICHERN.PHP4` öffnet eine bestimmte Datei – hier `HIGHSCORE.TXT` – und fügt dieser Datei die von Flash übergebenen Variablen ein. So kommt je Eintrag ein Wert für `name` und ein Wert für `hits` hinzu.

Und so funktioniert die Einbindung:

Du gehst in den „GameOver“-Frame und löschst den Button „Ab zum Gewinnspiel“. Nun hast du genug Platz für deinen Highscore Eintrag.

Mit dem Textwerkzeug erstellst du zunächst die Beschreibung, z.B. eine Überschrift wie *Highscore-Eintrag*; darunter zwei Zeilenbeschriftungen mit *Name:* und *Portionen:*. Neben den Text *Name:* fügst du ein weiteres Textfenster ein, das du unter der Rubrik `TEXTOPTIONEN` auf `TEXTEINGABE` stellst. Als `VARIABLE` gibst du `name` ein. Damit der Spieler weiß, dass er hier etwas eingeben kann, wählst du die `OPTION RAND` aus, so dass das Feld auch wie ein Textfeld aussieht.



Neben *Portionen:* fügst du ebenfalls ein Textfenster ein. Dieses stellst du unter `TEXTOPTIONEN` auf `DYNAMISCHER TEXT`; unter `VARIABLE` gibst du `gefressen` ein.



Vergiss nicht, die richtige Schriftfarbe auszuwählen, z.B. weiß. Die Farbe kannst du über das Fenster **TEXTOPTIONEN** unter der Rubrik **ZEICHEN** einstellen.

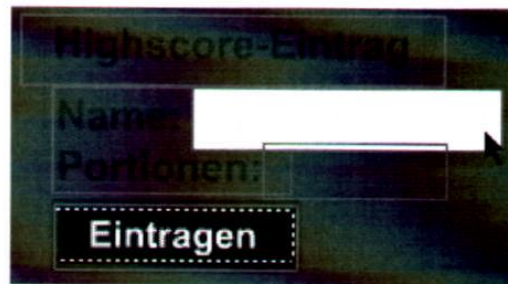
Fehlt noch ein Button mit der Aufschrift *Eintragen*. Diesen erstellst du so wie den Button „Ab zum Gewinnspiel“. Hierfür nutzt du das **RECHTECKWERKZEUG** und das **TEXTWERKZEUG**.

Am besten erstellst du den Button außerhalb der Bühne. Hier kannst du ungestört arbeiten – z.B. wird das Markieren nicht durch andere Objekte gestört.

Ist der Button fertig, markierst du ihn mit dem **PFEILWERKZEUG** und gruppierst ihn per **MODIFIZIEREN|GRUPPIEREN**. Danach wandelst du ihn per **EINFÜGEN|IN SYMBOL KONVERTIEREN** in ein Symbol um; als **VERHALTEN** wählst du hier **SCHALTFLÄCHE** und als **NAME** vergibst du *eintragen*. Klickst du nun diesen Button doppelt an, gelangst du in den Bearbeitungsmodus für diese Schaltfläche. Mit der rechten Maustaste klickst du hier auf den Frame **DRÜBER** und wählst im Menü **SCHLÜSSELBILD EINFÜGEN**. Klickst du dann den Text im Button doppelt an, kannst du z.B. die Farbe des Textes ändern, so dass sich später ein Effekt zeigt, wenn der Spieler mit der Maus über den Knopf fährt.

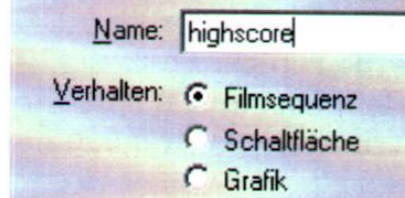
Per Doppelklick auf **SZENE 1** gelangst du wieder in den Hauptfilm zurück. Als nächstes soll folgendes geklärt werden: Klickt der Spieler auf den Button, soll das Programm **SPEICHERN.PHP4** ausgeführt werden. Ist der Eintrag gespeichert, dann soll die Datei

LADEN.PHP4 ausgeführt werden, und die Variablen **Namen** und **Hits** sollen in zwei Textfenstern angezeigt werden. Damit das alles vernünftig koordiniert wird, solltest du aus dem kompletten Highscore-Eintrag eine eigene Filmsequenz erstellen. So erhält dieser Bereich eine eigene Zeitleiste, und du kannst die **ACTIONSCRIPTE()** besser koordinieren. Für die Umwandlung in ein Symbol musst du alle Objekte markieren. Hierfür hältst du die **UMSCH**-Taste gedrückt und klickst mit der linken Maustaste nach und nach alle Objekte des Eintrages an, also jeden Text, die Textfelder und den Button.

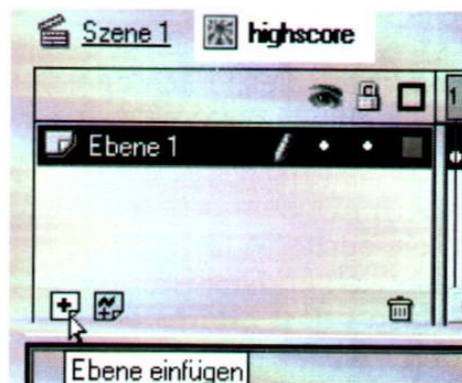


Dann wandelst du das Markierte über **EINFÜGEN|IN SYMBOL KONVERTIEREN** um. Als **NAME** vergibst du z.B. *Highscore*, als **VERHALTEN** wählst du **FILMSEQUENZ**.

Symboleigenschaften



Klickst du nun diese Filmsequenz doppelt an, gelangst du in den Bearbeitungsmodus für *highscore*. Hier fügst du per Klick auf den **+**-BUTTON eine neue Ebene ein.



In diese neue Ebene gehören die `ACTIONSCRIPT()` für das Speichern und Auslesen der Datei **HIGHSCORE.TXT**. Zunächst klickst du mit der rechten Maustaste auf den Frame 1 in der neuen Ebene und wählst **AKTIONEN**.

Am besten stellst du gleich den Expertenmodus ein!

Die PHP-Skripte von **Sebastian Wichmann** greifen nicht direkt auf **HIGHSCORE.TXT** zu – also musst du den Dateinamen aus Flash übergeben. Hierzu legst du eine Variable mit dem Namen `datei` an. Der Inhalt dieser Variable ist `"highscore.txt"`. Die Zeile heißt:

```
var datei="highscore.txt";
```

Als nächstes musst du verhindern, dass Flash diese Filmsequenz sofort beim Aufruf abspielt. Flash muss schließlich darauf warten, dass der Spieler seinen Namen eingibt und auf *Eintragen* klickt. Hierfür nutzt du den `stop()`; **Befehl**. Also steht in der zweiten Zeile:

```
stop();
```

Gibt nun der Spieler seinen Namen ein, drückt er entweder die **ENTER-TASTE** oder klickt mit der Maus auf den Button *Eintragen*. Also musst du ein `ACTIONSCRIPT()` für die Abfrage der **ENTER-TASTE** hinterlegen sowie für die **MAUSTASTE**.

Da sich der Mausklick auf den Button bezieht, hinterlegst du auch hier das `ACTIONSCRIPT()`. Hierfür klickst du mit der rechten Maustaste auf den gerade erstellten *Eintragen*-Button und wählst im Menü **AKTIONEN**.

Nun kommt ein `ACTIONSCRIPT()`, das beim Mausklick oder beim Betätigen der **ENTER-TASTE** zum Frame 2 dieser Filmsequenz springen soll. Wie schon bei den anderen Buttons nimmst du hierfür die Abfrage `on(release)`, was soviel bedeutet wie „Beim Loslassen der Maustaste“.

```
on (release) {..Befehle...}
```

In diese Abfrage kannst du auch gleich die Tastenabfrage für die **ENTER-TASTE** legen, und zwar per `keyPress"<Enter>"`. Das ganze sieht dann so aus:

```
on (release, keyPress"<Enter>"; { gotoAndPlay(2); }
```

Mit dem `,` trennst du in der Klammer die Bedingungen, die notwendig sind, damit Flash die in den Klammern stehenden Befehle ausführt.

Das `,` ist eine **ODER**-Anweisung und keine **UND**-Anweisung!

Nun kommt folgende Abfolge:

- 1.) Übergabe des Namens und des Punktestandes an die Datei **SPEICHERN.PHP4**
- 2.) Aufruf der Datei **LADEN.PHP4**, damit die Highscoreeinträge an Flash übergeben werden
- 3.) Wenn die Inhalte geladen sind, soll Flash sie anzeigen

Die von dir eingefügte Ebene 2 ist deine Scriptebene; in Ebene 1 verwaltest du die Texte und Grafiken. In der Ebene 2 fügst du in Frame 2 ein leeres Schlüsselbild ein, indem du du mit der rechten Maustaste auf diesen Frame klickst und den entsprechenden Menüpunkt wählst.

Dann klickst du wiederum mit der rechten Maustaste auf diesen Frame und klickst auf **AKTIONEN**. Nun folgt das `ACTIONSCRIPT()` für die Übergabe der Variablen an **SPEICHER.PHP4**. Es gibt den `ACTIONSCRIPT-BEFEHL` `loadVariablesNum(datei, level, post/get)`.

Mit diesem Befehl kannst du Variablen entweder laden oder senden. Je nachdem, was du als letzte Option eingibst, sendet Flash die Variablen (**Post**) oder holt sich die Variablen (**Get**). In diesem Fall sieht die Scriptzeile so aus:

```
loadVariablesNum
("speichern.php4?datei="+this.datei+"&name="+this.name+"&hits="+_root.gefressen, 0, "POST");
```

Im Einzelnen:

```
"speichern.php4?datei="+this.datei
+"&name="+this.name+"&hits="+_root
.gefressen
```

Dies teilt mit, welche Datei aufgerufen wird. Werden PHP Dateien aufgerufen, können per ? Variablen übergeben werden. Hierzu nennst du einfach den Namen der Variablen, und nach dem =-Zeichen folgt ihr Wert. Mit dem &-Zeichen teilst du PHP mit, dass noch weitere Variablen folgen.

In diesem Fall übergibst du die Variablen `datei`, `name` und `hits`. Die Zeile sieht ein wenig unübersichtlich aus, weil du einen Teil des Dateiaufrufes als Text eingibst und den anderen Teil aus den Variablen in Flash beziehst. Die Dinge, die du statisch hast, wie z.B. Dateiname und Variablennamen, gibst du mit "" ein; die Dinge, die aus Flash kommen, musst du zwischen den "" per + einfügen.

```
"speichern.php?datei="+variable+
&name="+variable+"&hits="+variable
...
```

Die Variablen selbst sprichst du flashtypisch an, also mit ihrer Instanz und ihrem Namen. Die Variable `Name` liegt in dieser Filmsequenz, also heißt es hier `this.name`. Die Variable `datei` liegt in dieser Filmsequenz, also heißt es hier `this.datei`. Die Variable `gefressen` liegt im Hauptfilm und deswegen heißt es hier `_root.gefressen`.

1.) , 0,

Hier erfolgt in der Syntax von `loadVariablesNum` immer eine Angabe zum Level, auf dem die Variablen verarbeitet werden müssen. Die 0 steht also für `_level0`.

2.) , "Post"

Hier gibst du an, was Flash machen soll – Versenden oder Abholen von Variablen. Post steht für Versenden, GET für Abholen.

Damit ist das Füllen der Datei `HIGHSCORE.TXT` erledigt. Als nächstes soll Flash die Datei auslesen und anzeigen.

Zunächst klickst du in der Ebene 1 auf den Frame 3 und wählst per rechter Maustaste den Menüpunkt `LEERES SCHLÜSSELBILD EINFÜGEN`. Nun sind das Eingabefeld und die Beschriftung des Highscoreeintrages nicht mehr zu sehen, und du kannst mit dem `TEXTWERKZEUG` den Text *"Highscoreliste wird geladen!"* einfügen.

Je nach Ladegeschwindigkeit kann das Auslesen der Datei ja länger dauern – da macht es Sinn, dass du dem Spieler mitteilst, was passiert. Nun klickst du mit der rechten Maustaste in Ebene 2 auf den Frame 4 und wählst ebenfalls `LEERES SCHLÜSSELBILD EINFÜGEN`.

Anschließend klickst du diesen Frame ein weiteres Mal mit der rechten Maustaste an und gelangst per Klick auf `AKTIONEN` in das Eingabefenster für `ACTIONSCRIPT()`. Das PHP Script von **Sebastian Wichmann** liefert an Flash eine Variable mit dem Namen `geladen`.

Als Wert übergibt er hier eine 1. Somit kannst du abfragen, ob der Ladevorgang beendet ist. Schließlich gibt es in diesem Flashfilm keine Variable mit diesem Namen.

Ist also die Bedingung erfüllt, dass `geladen==1` ist, geht das nur, wenn das PHP-Script `LADEN.PHP4` bereits ausgeführt ist. Würdest du hierzu keine Abfrage einbauen, könnte es passieren, dass die Highscoreliste ohne Inhalt angezeigt wird. Folgende Abfrage kommt also nun in den Frame 4:

```
Wenn (die Variable geladen == 1)
dann
```

```
{soll Flash diese Filmsequenz
weiter abspielen}
```

```
sonst
```

```
{soll Flash die Datei
laden.php4?datei aufrufen und
zurück in Frame 3 springen}
```

Das Script sieht dann so aus:

```
if (geladen == "1") {
play ();
} else {
```

```
loadVariables("laden.php4?datei="+
this.datei+"&" + random(99999) ,
this, "GET");
gotoAndPlay (3);
}
```

Hier gibt es nun einen Unterschied zum speichern. Während der Befehl `loadVariablesNum()` die Variablen in oder aus der Hauptzeitleiste liest, arbeitet das `loadVariables()` nur mit der aktiven Filmsequenz.

Der Aufruf der Datei funktioniert genauso wie vorher. Du verbindest statische Information mit Variablen von Flash per `""` und den `+-Zeichen`.

Da die Internetbrowser dazu neigen, aus dem Cache zu lesen, hängst du hier noch eine per Zufall generierte Zahl an. Diese hat für die Ausführung der Datei `laden.php4` keine Bedeutung – aber dies gibt dem Aufruf immer eine andere Aufrufadresse und verhindert so den Browser, die Datei aus dem Cache zu holen.

So ruft diese Zeile z.B.

LADEN.PHP4&444 auf und beim nächsten mal **LADEN.PHP4&1287**. Da du als Sendeoption nun GET einsetzt, weiß Flash, dass dieses Mal Variablen geladen werden. Diese Variablen sind **geladen=1, namen=liste der gespeicherten Namen und hits=Liste der gespeicherten Spielstände**.

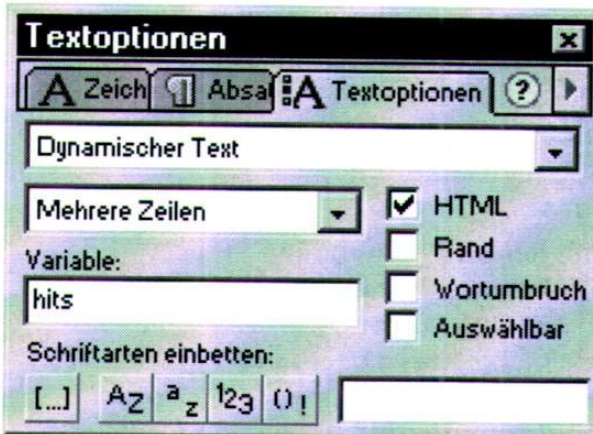
Als nächstes fügst du per rechter Maustaste und entsprechender Wahl im Menü in Frame 5 in Ebene 1 ein LEERES SCHLÜSSELBILD ein.

Hier gibst du nun mit dem Textwerkzeug eine Überschrift ein, z.B.

"Highscoreliste", und darunter die Spalten Überschriften *"Portionen:"* und *"Namen:"*.

Nun fehlen noch zwei dynamische Textfelder. Für jede der Überschriften "Portionen" und "Namen" zeichnest du ein eigenes Textfenster. Diese beiden Fenster sollten die gleiche Höhe haben. Eines verbindest du mit der Variable **namen** und das andere mit der Variable **hits**.

Da **Sebastian Wichmann** mit dem HTML-Befehl `
` nach jedem Eintrag einen Zeilenumbruch generiert, musst du im ZEICHEN-MENÜ unter TEXTOPTIONEN den Reiter HTML markieren.



Achte darauf, dass diese zwei Textfelder von keinem anderen Textfeld überdeckt oder berührt werden! Sonst kann es passieren, dass sie nicht angezeigt werden!

In der Ebene 2 fügst du im Frame 5 ebenfalls ein LEERES SCHLÜSSELBILD ein. Hier hinterlegst unter AKTIONEN den ACTIONSCRIPT()-BEFEHL `stop()`; . So verhinderst du, dass Flash diese Filmsequenz wieder von vorne beginnt. Fertig.

Achte darauf, dass du zum Test dieses Spielstands eine Serversoftware und einen PHP-Interpreter installiert haben musst. Hast du das nicht, kannst du diesen Stand nur online testen.

Erweiterungen

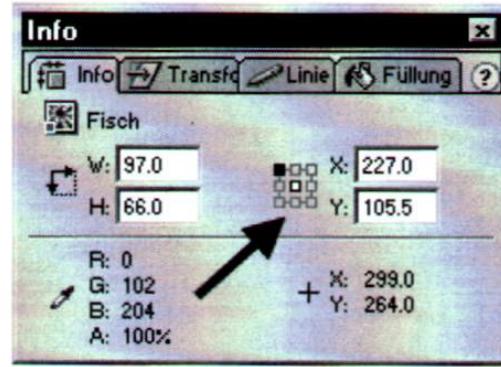
Warum schwimmt Fishbud aus dem Aquarium?

Es folgen nun einige Erweiterungen, die du am Spiel vornehmen kannst. Schwimmt z.B. Fishbud über den Aquariumrand hinaus, ist das zwar nicht weiter schlimm – aber wer es genau nimmt, der will, dass er am Rand stehen bleibt. Im Augenblick sieht das Programm so aus:

```
onClipEvent (keyDown) {
    // nach oben
    if (Key.getCode() == 38) {
        _level0.Fisch._y -= 3;
    }
    // nach unten
    if (Key.getCode() == 40) {
        _level0.Fisch._y += 3;
    }
    // nach links
    if (Key.getCode() == 37) {
        _level0.Fisch.gotoAndStop(2);
        _root.richtung="links";
        _level0.Fisch._x -= 3;
    }
    // nach rechts
    if (Key.getCode() == 39) {
        _root.richtung="rechts";
        _level0.Fisch.gotoAndStop(1);
        _level0.Fisch._x += 3;
    }
    // Leertaste
    if (Key.getCode() == 32) {
        if (_root.richtung=="rechts") {
            _level0.Fisch.gotoAndStop(4);
        }
        else {
            _level0.Fisch.gotoAndStop(3);
        }
    }
}
```

In den Fällen wo du die X oder Y Koordinate veränderst, musst du nun eine Abfrage einbauen, die prüft, ob der entsprechende Wert seinen Maximalwert erreicht hat.

Hierfür musst du wissen, dass die x-y-Koordinaten deines Objektes über das INFO-FENSTER erkennbar sind. In der Mitte von Fishbud siehst du ein kleines Kreuz. Auf dieses Kreuz beziehen sich die x-y-Koordinaten.

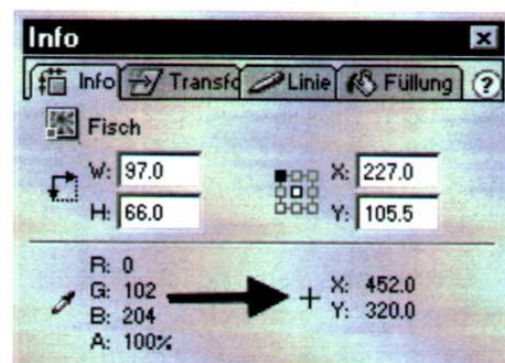


Neben den Koordinaten siehst du ein Quadrat-Raster. Hier kannst du entweder den Mittelpunkt wählen oder die linke obere Ecke.

Unser Film hat die Maße 550 x 450; Fishbud selbst ist 97 x 66 Pixel groß. Schwimmt Fishbud nach unten, musst du neben dem Abstand zum Beckenboden auch die eigene Höhe von Fishbud beachten, also die 66 Pixel.

Da Flash vom Mittelpunkt ausgeht, musst du für die Bestimmung der Maximalkoordinaten die Höhe und die Breite von Fishbud durch zwei teilen. In der Höhe musst du also 33 Pixel für Fishbud berechnen, in der Breite 48 Pixel.

Für die Ermittlung der anderen Werte kannst du ebenfalls das INFO-FENSTER nutzen. Fährst du mit der Maus über die Bühne, siehst du hier die x-y-Koordinaten der Mausposition.



Die tiefste Stelle, die Fishbud erreichen soll, liegt bei 360 Pixel; ziehst du hiervon nun die 33 Pixel Eigenhöhe ab, weißt du, dass der y-Wert von Fishbud maximal auf 327 Pixel kommen darf.

Am oberen Rand darf er maximal den Wert von 44 Pixel erreichen. Hier musst du daran denken, dass die Höhe von Fishbud sich Höhe von 66 auf 88 Pixel erhöht, wenn er isst. Geteilt durch zwei

ergibt dies 44. So ergeben sich also für die y-Werte die Werte 44 für oben und 327 für unten. Für den x-Wert kannst du auf diesem Wege auch deine Maximumwerte ermitteln; ich komme hierbei auf 48 Pixel für den linken Beckenrand und 502 Pixel für den rechten Beckenrand.

Der Ort, an dem du diese Werte nun einbaust ist das `ACTIONSCRIPT()`, das Fishbud steuert. Du klickst ihn im Hauptfilm mit der rechten Maustaste an und wählst AKTIONEN. Die Programmlogik muss hier wie folgt erweitert werden.

```
Wenn eine Taste gedrückt wurde {
Wenn (die Taste "Pfeil nach oben"
gedrückt wurde und wenn der y-Wert
von Fishbud größer ist als 48)
dann
{verschiebe Fishbud um 3 Pixel
nach oben}
Wenn (die Taste "Pfeil nach unten"
gedrückt wurde und wenn der y-Wert
von Fishbud kleiner ist als 326)
dann
{verschiebe Fishbud um 3 Pixel
nach unten}
Wenn (die Taste "Pfeil nach links"
gedrückt wurde und wenn der x-Wert
von Fishbud größer ist als 48)
dann
{verschiebe Fishbud um 3 Pixel
nach links}
Wenn (die Taste "Pfeil nach oben"
gedrückt wurde und wenn der x-Wert
von Fishbud kleiner ist als 502)
dann
{verschiebe Fishbud um 3 Pixel
nach rechts}
}
```

In `ACTIONSCRIPT()` sieht die **UND Verknüpfung** so aus:

```
if (Key.getCode() == 38 &&
_level10.Fisch._y>44)
{ _level10.Fisch._y -= 3; }
```

Innerhalb der **IF-Abfrage** hängst du mit dem **&&** eine **UND** Bedingung an. Der Inhalt der nachfolgenden Klammer wird nur dann ausgeführt, wenn beide Bedingungen erfüllt sind. Mit einem **||** erhältst du eine **ODER-Abfrage**. In die anderen drei Tastaturabfragen für die Bewegung werden folgende Ergänzungen eingesetzt:

```
_level10.Fisch._y<326
_level10.Fisch._x>48
_level10.Fisch._x<502
```

Bei der Erweiterung für links und rechts musst du beachten, dass innerhalb dieser Abfrage auch das „Drehen“ von Fishbud erfolgt. Obwohl Fishbud sich im Rande befindet, muss er sich drehen können. Deswegen darfst du hierbei nicht mit der **&&-Verknüpfung** arbeiten, sondern du musst eine separate **If-Abfrage** programmieren. Die Programmlogik für „Links“ lautet:

```
Wenn die Taste Nr. 37 gedrückt
wurde
dann {
Lade den nach links schwimmenden
Fishbud;
Setze die Variable Richtung auf
"links";
Wenn die x-Koordinate von Fishbud
größer ist als 48
dann{
Verschiebe Fishbud um 3 Pixel nach
links;
}
}
```

In `ACTIONSCRIPT()` sieht die Änderung dann so aus:

```
if ( _level10.Fisch._x>48){
_level10.Fisch._x -= 3;}
```

Geschafft - nun hat Fishbud den Aquariumrand als Begrenzung für seine Bewegung. Allerdings fällt nun folgendes auf: Ist die Runde überstanden, und Flash springt wieder auf Frame 2, dann steht Fishbud plötzlich wieder auf seiner Startposition. Dies liegt an dem Schlüsselbild in Frame 170 auf Ebene 1.

Lösche es einfach, indem du es mit der rechten Maustaste anklickst und BILDER LÖSCHEN wählst.

Hast du das Schlüsselbild in Frame 170 gelöscht, musst du den Tetrisbalken von Fishbud von Frame 169 auf den Frame 170 verlängern. Dazu klickst du diesen Frame an, hältst die linke Maustaste gedrückt, gehst mit der Maus auf den Frame 170 und lässt die Maustaste los.

Level mit unterschiedlichen Schwierigkeitsgraden

Die Idee ist nun folgende: Es soll nach einer bestimmten Zeit einen neuen Schwierigkeitsgrad geben, z.B. in dem sich die Menge der Portionen erhöht. Du könntest hierfür z.B. drei Futterwellen vorsehen.

Sind die drei Futterwellen überstanden, zählst du eine Variable hoch, die die Menge der Portionen festlegt, und beginnst wieder von vorne. So hast du automatisch einen höheren Schwierigkeitsgrad.

Für diese drei Futterwellen nutzt du entweder ACTIONSCRIPT(), oder du hängst an die jetzige Futterwelle einfach zwei weitere an. Da ich vorher versprochen habe, dass das Aneinanderhängen von Tetrisbalken noch kommt, mache ich nun mit der zweiten Variante weiter, dem Verlängern der Hauptzeitleiste.

Soll das Futter nicht nur einmal fallen, sondern dreimal, bevor Flash wieder von Frame 2 beginnen soll, verlängert sich der Zeitplan von Frame 170 auf "170+160+160" = Frame 490 + 1 Frame für Zusatzscripte=491.

Läuft der Film bis zum Frame 491, erfolgt in Frame 492 der `gotoAndPlay(2);`-Befehl. Alle anderen Ebenen verschiebst du entsprechend nach hinten. Am besten beginnst du mit der Anpassung der Scriptebene bzw. der Produktion der Futterportionen. Du übernimmst nun die Programmierung aus Frame 10 in die Frames 172 und 331.



Als erstes musst du das ACTIONSCRIPT() für die Korrektur der Variablen **verloren** umsetzen. Nach dem Fall der ersten Welle muss also nun ein ACTIONSCRIPT() kommen, das prüft, ob die Runde 0 läuft, und die Variable **verloren** entsprechend korrigiert.

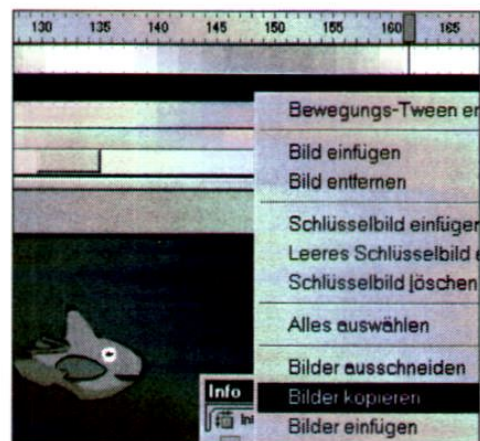
Dieses Script erfolgt im Frame 171. Hierfür klickst du mit der rechten Maustaste in Ebene 2 auf den Frame 171. Nach einem Klick auf LEERES SCHLÜSSELBILD EINFÜGEN und danach auf AKTIONEN gelangst du in den Programmiermodus.

Am besten wieder auf Expertenmodus stellen!

Hier erfolgt folgende Abfrage:

```
if (_root.runde==0) {
    _root.verloren--;
}
```

Anschließend gehst du auf den Frame 2 und löschst das hier liegende ACTIONSCRIPT(). Nun erstellst du die weiteren Futterwellen: Du kopierst den Scriptbereich des Futters, indem du mit der linken Maustaste auf den Framebereich zwischen 10 und 170 klickst, so dass dieser Filmbereich schwarz markiert wird, und dann mit der rechten Maustaste auf den markierten Bereich klickst und BILDER KOPIEREN wählst.



Nun klickst du mit der rechten Maustaste auf den Frame 172 und wählst **BILDER EINFÜGEN**.

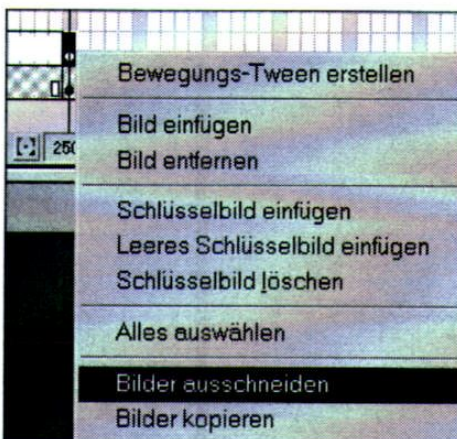
Du siehst unmittelbar, dass die Ebene 2 nun bis zum Frame 331 reicht. Klickst du nun direkt mit der rechten Maustaste auf den Frame 332 und wählst wieder **BILDER EINFÜGEN**, reicht diese Ebene bis zum Frame 491. In Frame 170 steht im Augenblick noch der Aufruf `gotoAndPlay(2);`.

Dieser Aufruf gehört nun in den Frame 492, ebenso wie das Hochzählen der Variablen `_root.runde`. Zuerst klickst du mit der rechten Maustaste auf den Frame 170 und löschst das **ACTIONSCRIPT()**. Dann klickst du mit der rechten Maustaste auf den Frame 492, fügst ein leeres Schlüsselbild ein und wählst **AKTIONEN**. Hier gibst du dann folgende Zeilen ein:

```
_root.runde++;  
gotoAndPlay(2);
```

Nun musst du die Tetrisbalken der anderen Ebenen bis zum Frame 492 verlängern.

Am besten beginnst du mit den Ebenen in denen das „Game Over“ eingebaut ist, also Ebene 1 und 4. Diese beiden Ebenen sollen von nun an im Frame 550 vorkommen. Du scrollst mit der Zeitleiste bis zum Frame 250. Hier klickst du mit der rechten Maustaste auf das Schlüsselbild in Ebene 1 und wählst **BILDER AUSSCHNEIDEN**.

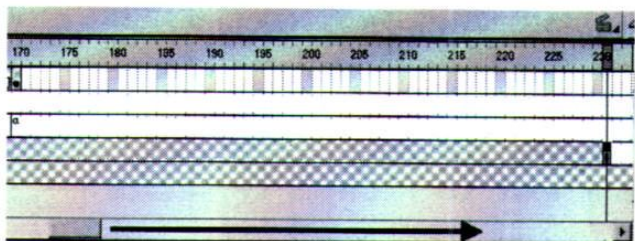
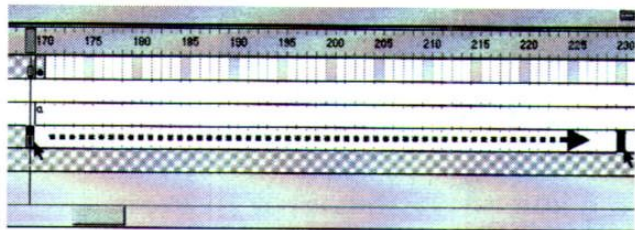


Dann scrollst du in der Zeitleiste weiter bis zum Frame 550, klickst hier den Frame 550 in Ebene 1 mit der rechten Maustaste an und wählst **BILDER**

EINFÜGEN. Dann scrollst du wieder zurück bis zum Frame 170. Hier löschst du das Schlüsselbild in Ebene 1 per rechter Maustaste und Klick auf **BILDER LÖSCHEN**.

Da in dieser Ebene unser Held Fishbud hinterlegt ist und wir dieses Objekt im ganzen Spiel benötigen, musst du den grauen Tetrisbalken in Frame 169 verlängern. Hierfür klickst du auf den Endframe und hältst die linke Maustaste gedrückt.

Nun kannst du den Balken durch die Mausbewegung nach rechts verlängern. Um über den Sichtbereich hinaus zu kommen, musst du den Balken loslassen und die Frameansicht weiter nach hinten verschieben. So verlängerst du Zug um Zug diesen Balken und zwar bis zum Frame 492.



Entsprechend gehst du mit den Endframe in Ebene 4 vor, nur dass du diesen bis zum Frame 550 verlängerst.

Den Tetrisbalken von Ebene 4 musst du bis zum Frame 550 verlängern, da diese ja auch im „Game Over“-Bild zu sehen sein soll!

Da nun der „Game Over“-Bereich nicht mehr in Frame 250 liegt, musst du das **ACTIONSCRIPT()** ändern, das den „Game Over“-Bildschirm aufruft. Dieses **ACTIONSCRIPT()** liegt im letzten Frame des Fischfutters *futterani*

Hierfür scrollst du in der Hauptzeitleiste zurück auf Frame 1. Das Futter liegt neben der Bühne. Du klickst es doppelt

an, so dass du in den Bearbeitungsmodus von *Futterani* kommst. Hier scrollst du bis zum Frame 160 und klickst das letzte Schlüsselbild mit der rechten Maustaste an. Per Klick auf AKTIONEN kommst du dann in den ACTIONSCRIPT() Bereich. Hier musst du den Frameaufruf von 250 in 550 ändern:

```
stop ();
_root.verloren++;
if (_root.verloren>5) {
_level10.gotoAndStop(550);
}
```

Als letzte Ebene musst du nun noch die Ebene 3 verlängern, denn hierin liegt das Futter – und das brauchen wir ja für die Futterwellen. Diese Ebene verlängerst du ebenso wie die anderen Ebenen per Maus Zug um Zug bis zum Frame 490.

Eigentlich bräuchtest du diese Ebene nur bis zum Frame 310 verlängern, da hier zum letzten Mal das Originalfutter gebraucht wird.

Nun kommt das neue Level hinzu. Hierzu musst du die Scripte, in denen das Futter geklont wird, ändern. In Ebene 2 Frame 10 findest du das erste. Hier gibt es die Schleife, die dreimal durchlaufen wird und jeweils eine Futterkopie einfügt.

```
for (var i=0; i<3 ; i++ )
```

Die Anzahl an Futterportionen wird durch den Vergleich von *i* und dem Wert 3 bestimmt. Tauschst du nun die 3 gegen eine **Variable** aus, bestimmt der Wert dieser **Variable** die Anzahl der Futterportionen.

Die neue Variable wird in Ebene 2 in Frame 1 zu den anderen Variablen hinzugefügt. Entsprechend erstellst du eine Variable, in der du den Wert für das Level speicherst.

```
var richtung = "rechts";
var gefressen=0;
var verloren =0;
var runde=0;
var anzahl=3;
var level=1;
```

Im Script änderst du den Schleifenaufruf entsprechend um:

```
for (var i=0; i<_root.anzahl; i++)
```

Du musst in allen drei Aufrufen diese Änderung der Schleife vornehmen, also auch im Frame 172 und 331.

Sind alle drei Scripte geändert, gehst du in das ACTIONSCRIPT() in Frame 491. Hier musst du nun die Werte der Variablen **level** und **anzahl** hochzählen.

```
_root.level++;
_root.anzahl++;
```

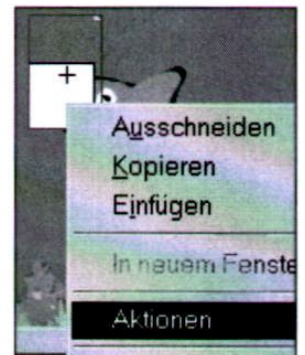
Diese Zeilen müssen über dem **gotoAndPlay(2);** stehen!

Nun ergibt sich ein Problem:

Da du per **hitTest()** die Kollision mit den Futterportionen prüfst, musst du hier weitere Abfragen für die zusätzlichen Futterportionen hinterlegen. Im Augenblick prüfst du nur bis zum **futter2**. Je nachdem wie gut der Spieler ist, kann es bis zum **futter7** gehen.

Da die Zählung bei 0 beginnt, ist **futter7** die achte Portion.

Hierfür gehst du in der Zeitleiste auf den Frame 1 und machst einen Doppelklick auf Fishbud. Hier klickst du auf den Frame 3. Nun klickst du mit der rechten Maustaste auf den Hitarea-Bereich und wählst AKTIONEN.



Am besten stellst du direkt den Expertenmodus ein!

Hier markierst du nun die letzte Abfrageschleife mit der Maus und wählst BEARBEITEN|KOPIEREN.


```

onClipEvent (enterFrame) {
    this._visible = false;
    if (this.hitTest(_root.futter0)) {
        removeMovieClip(_root.futter0);
        _root.gefressen++;
        _level0.Fisch.gotoAndPlay(2);
    }
    if (this.hitTest(_root.futter1)) {
        removeMovieClip(_root.futter1);
        _root.gefressen++;
        _level0.Fisch.gotoAndPlay(2);
    }
    if (this.hitTest(_root.futter2)) {
        removeMovieClip(_root.futter2);
        _root.gefressen++;
        _level0.Fisch.gotoAndPlay(2);
    }
}

```

Dann gehst du mit dem Cursor vor die letzte Klammer und fügst per BEARBEITEN|EINFÜGEN bzw. per Tastenkombination STRG+V diese Schleife fünfmal ein. Nun änderst du in jeder der neu eingefügten Schleifen den Aufruf des `_root.futter x` in aufsteigender Reihenfolge von `_root.futter3` bis zu `_root.futter7`.

```

if (this.hitTest(_root.futter7)) {
    removeMovieClip(_root.futter7);
        _root.gefressen++;
    _level0.Fisch.gotoAndPlay(2);
}

```

Vergiss nicht, diese Änderungen auch im nach rechts fressendem Fishbud vorzunehmen, also im Frame 4 von Fishbud!

Da du hier die Anzahl der Portionen auf maximal 8 vorgibst, solltest du im Hauptfilm in der Scriptebene das Hochzählen der Variable Anzahl mit einer If-Abfrage umranden! Diese Erweiterung kommt in den Frame 545 der Scriptebene.

```

if (_root.anzahl<8) {_root.anzahl++;}

```

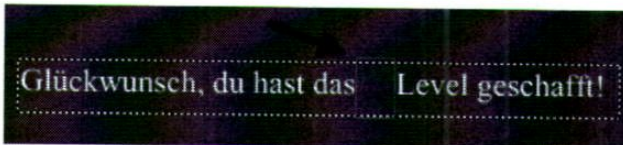
Damit der Spieler auch weiß, dass er in ein neues Level kommt, kannst du die Frames zwischen 491 und 550 für eine Levelanzeige nutzen. Zunächst ziehst du in Ebene 2 den Frame 491 mit der Maus auf den Frame 545. Danach fügst du in Ebene 1 auf Frame 492 ein leeres Schlüsselbild ein. Hierfür klickst du den Frame mit der rechten Maustaste an und wählst LEERES SCHLÜSSELBILD

EINFÜGEN. Nun kannst du mit dem TEXTWERKZEUG einen Text einfügen, z.B. *Glückwunsch, du hast das Level geschafft!*

Dann fügst du ein weiteres Textfenster ein, das du unter der Rubrik TEXTOPTIONEN als DYNAMISCHEN TEXT einstellst. Unter VARIABLE gibst du hier `level` ein.



Dieses Textfenster legst du über die "Lücke" in deinem Glückwunschtext.



Da Fishbud nun eine kurze Zeit nicht mehr zu sehen ist, ist er beim Aufruf des neuen Levels wieder auf seiner Startposition. Deswegen musst du die Variable `richtung` wieder auf den Standardwert `rechts` setzen. Dies machst du am besten in der Scriptebene auf Frame 545.

```

_root.runde++;
_root.level++;
if (_root.anzahl<8) {
    _root.anzahl++;}
_root.richtung="rechts";
gotoAndPlay(2);

```

Levelanzeige

In einem „echten“ Computerspiel wird normalerweise immer das Level angezeigt, in dem man sich befindet. Unter dem Aquarium ist noch genug freier Platz, so dass du hier noch eine Levelanzeige einbauen kannst. Hierfür klickst du in Ebene 4 auf den Frame 1.

Per Textwerkzeug fügst du hier den Text Level ein sowie ein leeres Textfenster mit der Einstellung DYNAMISCHER TEXT und dem Verweis auf die VARIABLE **level**.

Am besten arrangierst du die Levelanzeige so, dass am rechten Rand noch ein wenig Platz ist. Hier kannst du dann mit dem Textwerkzeug noch den Titel des Spieles "Fishbud" eintragen. So weiß der Spieler nicht nur in welchem Level er ist sondern auch welches Spiel er spielt!

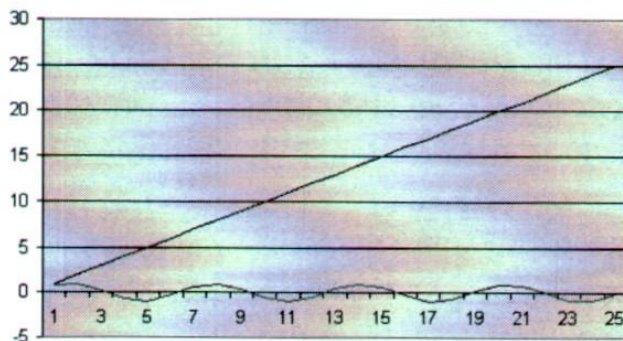
Fertig. Nun hast du einen dynamischen Spieleablauf mit bis zu acht Futterportionen sowie am Ende des Spiels eine Highscoreliste zum Eintragen. Diesen Stand kannst du dir auf

www.spieleinflash.de herunterladen. Die Datei heißt **FISHBUD7.FLA**.

Realistischer Futterfall

Wenn ein Fischfreund sich diesen Spielstand anschaut, wird er wohl bemängeln, dass Fischfutter im wirklichen Leben niemals gradlinig nach unten fällt. Eigentlich gleitet es in einer Schlangenlinie abwärts.

Um deinem Futter auch eine „Schlangenlinien“-Bewegung zu ermöglichen, benötigst du ein wenig Mathematik; so stellt z.B. der Graph der Sinusfunktion eine kontinuierliche „Schlangenlinie“ dar und ist für einen realistischen Futterfall eine gute Basis:



Die blaue Linie ist ein kontinuierlich zunehmender Wert, die rote Linie ist die aus diesem Wert resultierende Sinuswelle. Positionierst du also das Futter anhand einer Sinusberechnung ständig neu, fällt es in einer Schlangenlinie.

Das Futter hat zwei Koordinatenwerte, die du für die Anpassung nutzen kannst: die x- und die y-Koordinate.



Soll sich das Futter in einer Wellenform bewegen, musst du zu der x-Koordinate des Futters ständig den Wert einer Sinusberechnung hinzu addieren.



Da das Futter eine eigene Filmsequenz mit 160 einzelnen Frames ist, bietet sich für dieses ständige Ändern der x-Koordinate folgende ActionScript() Abfrage an:

```
onClipEvent (enterFrame) {Befehle}
```

Jedesmal wenn Flash den nächsten Frame der Futteranimation lädt, werden die Befehle in den geschwungenen Klammern ausgeführt. Addierst du hier zur x-Koordinate einfach das Ergebnis einer Sinusberechnung, erhältst du automatisch eine Schlangenlinienbewegung. Dieses Script soll als AKTION zum Hauptfutter eingesetzt werden.

Also gehst du in den Hauptfilm und klickst mit der rechten Maustaste auf das Futter. Mit einem Klick auf AKTIONEN gelangst du in den ACTIONSCRIPT() Modus.

Als ActionScript gibst du zunächst folgendes ein:

```
onClipEvent (enterFrame)
{...Befehle...}
```

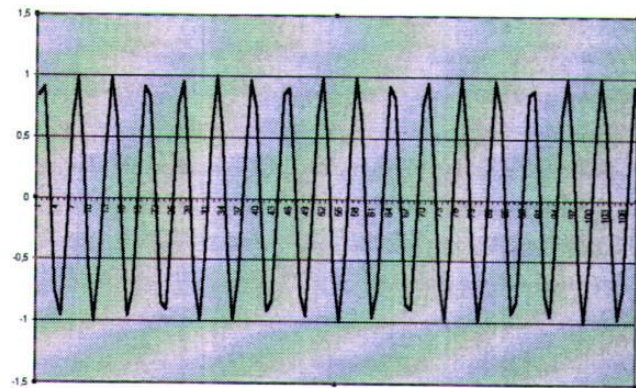
Damit die Sinuswerte sich auch bewegen, musst du einen Wert haben, der kontinuierlich zunimmt (siehe blaue Linie). Hierfür bietet sich eine Variable an.

Da dieses ACTIONSCRIPT() später auch in den Klonen laufen soll, darfst du hier keine allgemeingültige Variable definieren, also nicht im Frame 1 in Ebene 2, sondern du benötigst eine lokale Variable. Der Unterschied ist, dass diese Variable nur in dieser Sequenz einen Einfluss hat. Durch die Zeile **sinusbasis ++**; erledigst du gleich zwei Probleme auf einmal:

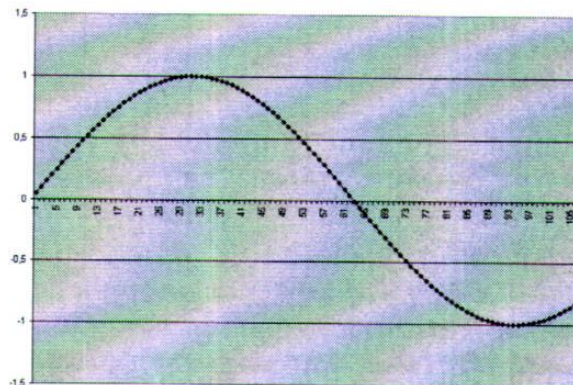
- 1.) Würdest du hier die Zeilen **var sinusbasis=0; sinusbasis++**; eintragen, würde die Variable immer beim Aufruf auf 0 gesetzt und dann hochgezählt werden. Im Resultat hättest du also keinen zunehmenden Wert.
- 2.) Diese eine Zeile kombiniert diese beiden Zeilen ineinander. Ist die Variable noch nicht da, wird sie von Flash erstellt. Existiert sie schon, wird der Wert einfach hochgezählt.

Nun zur Sinusberechnung. Die Werte einer Sinusberechnung bewegen sich in einem Zahlenraum von 1 bis -1. Begrenzt du die Berechnung auf ganze,

natürliche Zahlen, erhältst du eine sehr zackige Welle.



Damit das Futter angenehmer fällt, solltest du also die Schritte der **sinusbasis** nicht in ganzen Zahlen, sondern in kleineren Dezimalzahlen vornehmen. Dumm ist nur, dass du per **sinusbasis ++**; immer in ganzen Zahlen hochzählst. Deswegen musst du einen kleinen Trick anwenden: Du teilst einfach deine **sinusbasis** auf eine kleinere Dezimalzahl herunter, z.B. indem du durch 20 teilst. Automatisch erhält deine Sinuskurve eine viel feinere Einteilung.



Die obere und diese untere Grafik bearbeiten den gleichen Zahlenraum von 1 bis 110; da diese untere Kurve aber in ihrer **sinusbasis** auf ein Zwanzigstel heruntergebrochen ist, läuft sie erheblich smarter und ist damit die richtige Basis für deine Schlangenlinie.

Da die Werte der Sinusberechnung immer nur zwischen 1 und -1 liegen, ist die Koordinaten-veränderung für manche Einsätze nicht hoch genug. So auch in diesem Fall! Deswegen multiplizierst du einfach den Wert der Sinusberechnung mit 1.5 oder 2.

Ebenso kannst du die Form der Schlangenbewegung ändern, indem du den Wert änderst, durch den du teilst. Teilst du z.B. nur durch 10, ist die Schlangenlinie erheblich kürzer und dynamischer; je höher der Wert, durch den du teilst, desto behäbiger ist die Schlangenlinie.

In ACTIONSCRIPT() sieht die Berechnung so aus:

```
Math.sin(sinusbasis/20)*1.5);
```

Damit sich der x-Wert des Futters ändert, addierst du einfach das Ergebnis dieser Berechnung zum X-Wert hinzu.

```
_x +=  
(Math.sin(sinusbasis/10)*1.5);
```

Das komplette Script sieht dann so aus:

```
onClipEvent (enterFrame) {  
  sinusbasis ++;  
  _x +=  
  (Math.sin(sinusbasis/10)*1.5);  
}
```

Da es im Augenblick auch vorkommen kann, dass mehrere Futterportionen den gleichen Startwert für ihre x-Koordinate haben, verdecken sie sich gegenseitig.

Mit einem kleinen Trick kannst du diese Schlangenbewegung nutzen, um die Portionen nicht synchron, sondern asynchron zu pendeln. Wäre die **sinusbasis** nicht ein kontinuierlich gleich zunehmender, sondern ein per Zufall hochgezählter Wert, würde die Schlangenlinie bei jeder Futterportion ein wenig anders verlaufen.

Hierfür änderst du dein Script um die Generierung einer per Zufall generierten Zahl im Raum von 1-5 und addierst diese Zahl zur **sinusbasis** anstelle des einfachen Hochzählens.

```
sinuschange=random(4)+1;
```

```
sinusbasis+=sinuschange;
```

Jede der Futterportionen bekommt nun diese kleine Zufallsberechnung und erhält damit ihre "eigene" Schlangenlinie. Doppelte Futterportionen sind somit sichtbar, da sie sich nicht synchron bewegen sonder asynchron.

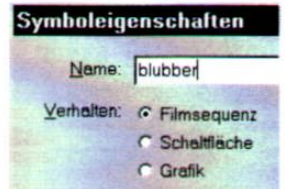
Bewegung im Aquarium

Ebenso wie der Fall des Futters würde der Fischfreund wohl auch die Luftpumpe vermissen, doch da können wir Abhilfe schaffen. Im Augenblick ist fürs Auge noch nicht viel los im Wasser. Einige sprudelnde Luftblasen sind hier von Nöten.

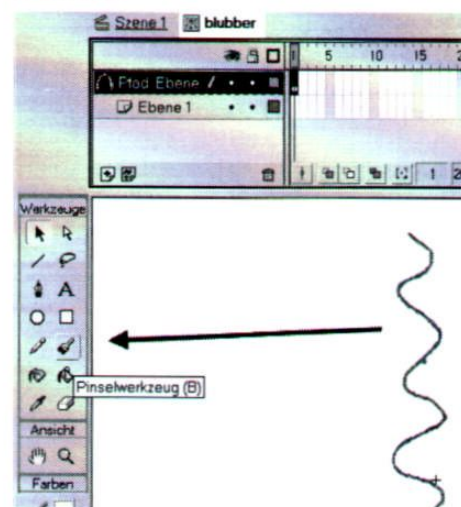
Hierfür fügst du zunächst über EINFÜGEN|NEUES SYMBOL ein neues Symbol ein.

Als VERHALTEN wählst du FILMSEQUENZ und als Namen vergibst du **blubber**.

Nun fügst du in dieser Filmsequenz eine Pfadebene ein, indem du auf PFAD-EBENE HINZUFÜGEN klickst.



Dann nimmst du dir das Pinselwerkzeug und malst eine Wellenlinie auf die Bühne.



Diese Wellenlinie ist der Pfad für deine Luftblasen, also sozusagen die Eisenbahnschiene, auf der die Luftblasen nach oben fahren sollen.

Als nächstes klickst du in der Pfadebene mit der rechten Maustaste auf den Frame 70 und wählst **SCHLÜSSELBILD EINFÜGEN**. Dann klickst du auf den Frame 1 in Ebene 1 und malst mit dem **ELIPSENWERKZEUG** einen kleinen Kreis.

Willst du einen Kreis zeichnen, hältst du die **UMSCH**-Taste gedrückt!

Die Randfarbe soll weiß sein und die Innenfarbe das Blau des Aquariumwassers haben.

Anschließend markierst du den gemalten Kreis mit dem Pfeilwerkzeug und gruppierst ihn per **MODIFIZIEREN|GRUPPIEREN**.



Dann wandelst du diesen Kreis in ein Symbol um, per Klick auf **EINFÜGEN|IN SYMBOL KONVERTIEREN**. Als Name vergibst du z.B. **Blase**, und als **VERHALTEN** wählst du

FILMSEQUENZ. So kannst du später dieser Blase noch verschiedene Effekte zuweisen. Effekte kann man nämlich nur Symbolen zuweisen!

Bewegst du nun den Kreis mit der Maus, siehst du sowohl den Umkreis als auch einen kleineren Kreis. Dieser kleinere Kreis ist zur Verankerung auf dem gemalten Pfad gedacht. Führest du den Kreis an das untere Ende des Pfades, siehst du, wie der kleine Kreis auf dem Pfad einrastet.

Nun klickst du mit der rechten Maustaste auf den Frame 35 und wählst **SCHLÜSSELBILD EINFÜGEN**. In diesem Schlüsselbild führst du den Kreis zum oberen Ende des Pfades.

Achte darauf, dass der Kreis wirklich einrastet!

Klickst du nun mit der rechten Maustaste auf den grauen Tetriskalken der Ebene 1 und wählst **BEWEGUNGS-TWEEN ERSTELLEN**, generiert Flash automatisch die Animation der Luftblase entlang des Pfades. Als Effekthascherei klickst du den Frame 35 in Ebene 1 an. Hier klickst du mit der rechten Maustaste die Luftblase und wählst **BEDIENFELDER|EFFEKT**. Unter diesem Menüfenster gibst du als Alpha-Wert 0 ein.

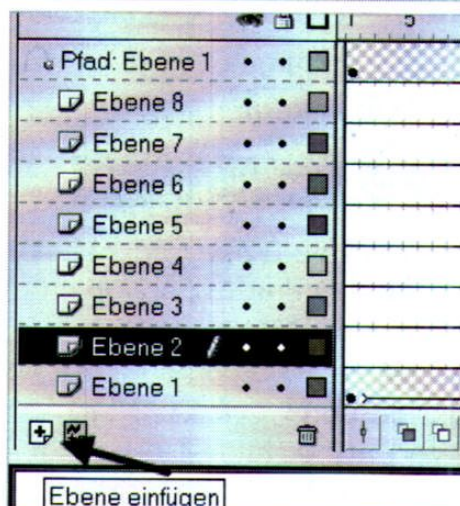


Hierdurch wird die Luftblase auf ihrem Weg nach oben ganz langsam unsichtbar.

Alphawert=100 = sichtbar
Alphawert=0=unsichtbar

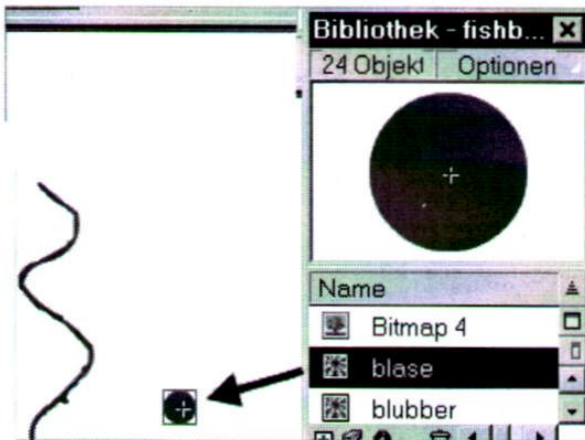
So eine Luftpumpe oder Umweltpumpe macht in der Regel mehr als eine Luftblase. Deswegen fügst du nun weitere Ebenen ein. Ich schlage die Anzahl 8 vor, d.h. du klickst siebenmal auf den Button **EBENE EINFÜGEN**.

Nur siebenmal, da du ja schon eine Ebene hast!



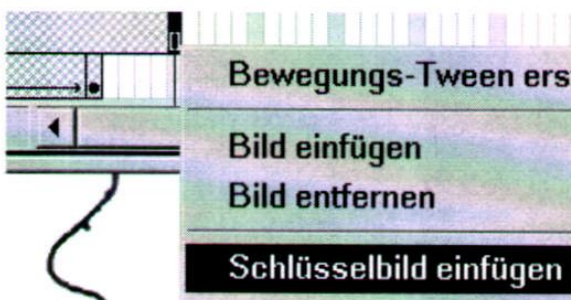
Nun klickst du mit der rechten Maustaste in Ebene 2, bzw. in der nächsten Ebene je nach dem von Flash vergebenen Namen, auf den Frame 5 und wählst **SCHLÜSSELBILD EINFÜGEN**.

Steht das Fenster BIBLIOTHEK nicht offen, dann öffnest du es per FENSTER|BIBLIOTHEK. Hier suchst du dir das Symbol **blase** und ziehst es per Drag&Drop mit der Maus auf die Bühne.



Diese *blase* positionierst du nun am unteren Ende deines Pfades. Übrigens siehst du hier, wo sich die erste *blase* gerade befindet.

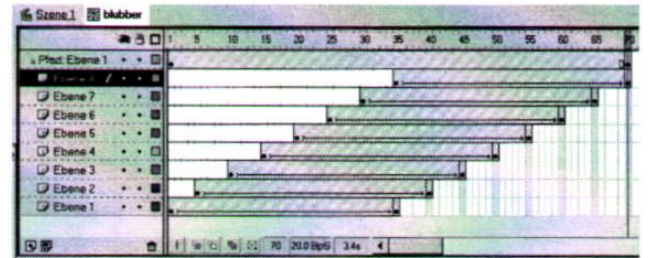
Flash hat übrigens automatisch einen Tetrisknoten für diese Ebene eingefügt. Ziehe mit der Maus den Endframe zurück auf den Frame 40. Klicke diesen Frame mit der rechten Maustaste an und wähle SCHLÜSSELBILD EINFÜGEN.



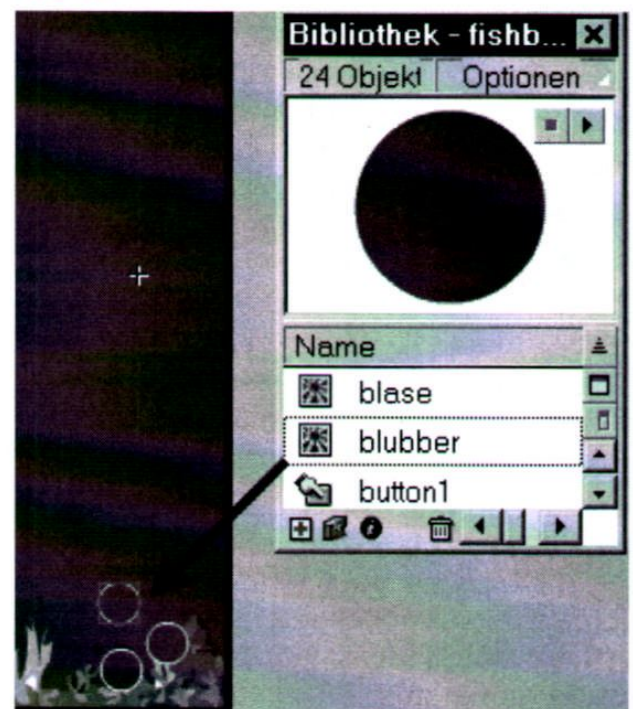
Nun nimmst du dir die Blase und ziehst sie bis zum Ende des Pfades. Achte darauf, dass sie auch tatsächlich einrastet!

Danach stellst du unter EFFEKT den ALPHA-Wert auf 0. Klickst du nun mit der rechten Maustaste auf den Tetrisknoten in dieser Ebene und wählst BEWEGUNGS-TWEEN ERSTELLEN, hast du die zweite deiner Luftblasen generiert.

Nun kommt ein wenig Fleißarbeit: Du musst nun nach und nach die Ebenen mit genau der gleichen Prozedur um eine Luftblase erweitern. Am Ende sollte deine Ebenenübersicht so aussehen.



Nun kehrst du durch einen Klick auf SZENE 1 in den Hauptfilm zurück. Hier klickst du die Ebene 4 an, also die Ebene, in der du das Aquarium und die Punktezahl hinterlegt hast. Hier kannst du nun über das Fenster BIBLIOTHEK so viele *blubber* einfügen wie du willst.



Fertig. Testest du nun den Film, siehst du die Luftblasen deiner Umweltpumpe im Wasser sprudeln!

Sound

Luftblasen machen Lärm

Da eine Umweltpumpe auch eine Menge Lärm macht, solltest du noch Sound hinterlegen. Als Basis kannst du hierfür z.B. **AIFF**-, **MP3**- oder **WAV**-Dateien verwenden. Im Internet findest du eine Menge an solchen Dateien.

Beachte bitte, dass es nicht immer unproblematisch ist, ohne ausdrückliche Erlaubnis einfach irgendwelche Töne von Websites wie z.B.

www.flashkit.com/soundfx/ zu verwenden. Das kann eventuell teuer werden!

Für die Soundunterstützung habe ich auf obiger Website drei Freeware Dateien gefunden. Diese Dateien sind

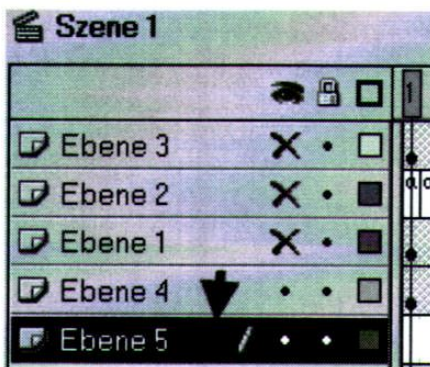
BUBLES.MP3 – von **dj guy-chi**

SCHLING.WAV – von **public domain**

DEPP_SPL_SITH_MAS-505.WAV von **Sith Master**

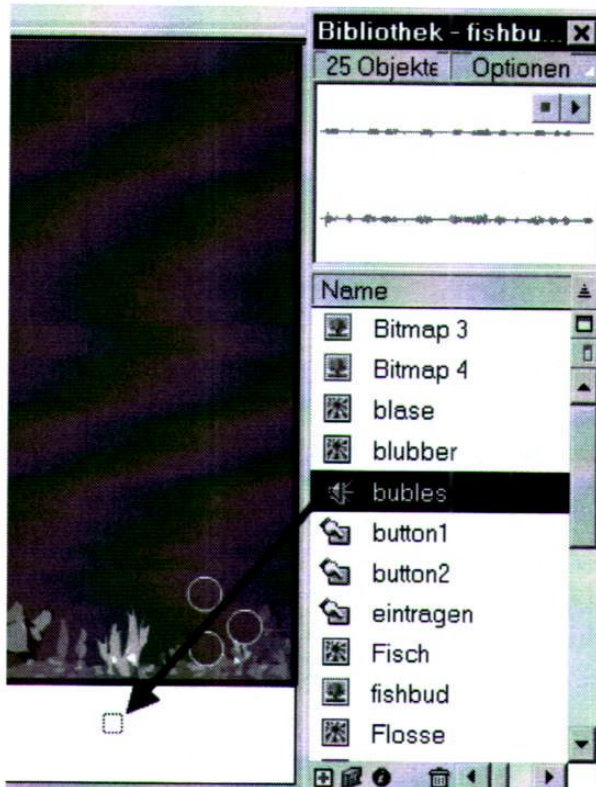
Diese Dateien stehen dir auch unter www.spieleinflash.de zum Download zur Verfügung.

Zunächst fügst du für den Grundsound deiner Umweltpumpe eine neue Ebene ein. Hierfür klickst du einfach auf den Button **EBENE EINFÜGEN**. Die neue Ebene ziehst du dann mit der Maus an die unterste Stelle in deine Ebenenübersicht.

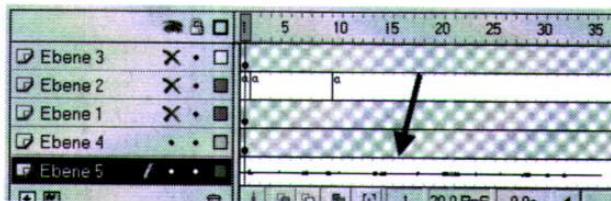


Nun importierst du die Sounddateien in diesen Flashfilm. Im Resultat stehen dir diese Sounds in der Bibliothek zur Verfügung, und du kannst sie einfach per Drag&Drop in deinen Film einfügen. Zunächst klickst du den Frame 1 in der

neuen Ebene 5 an. Dann öffnest du das **BIBLIOTHEK-FENSTER** und suchst die Sequenz *bubbles*, die du per Drag & Drop auf die Bühne des Films ziehst:

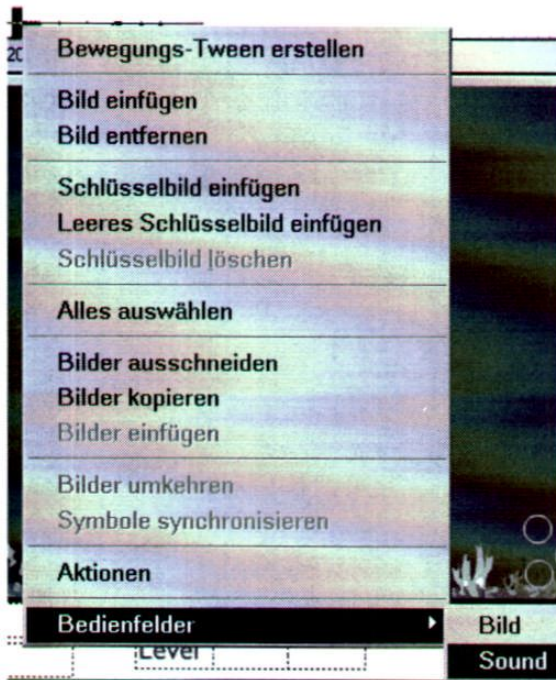


Als Ergebnis siehst du dann, dass Flash in der Zeitleiste eine Wavekurve eingefügt hat.



Im Augenblick dauert der Sound bis zum Frame 36. Allerdings ziehst du sicher vor, dass das Blubbern die ganze Zeit anhält – also musst du das Flash mitteilen, indem du die Welle mit der rechten Maustaste anklickst und im Menü **BEDIENFELDER | SOUND** wählst.

In dem Menü kannst du verschiedene Einstellungen vornehmen. Unter **EFFEKT** kannst du Dinge wie **EINBLENDEN**, **AUSBLENDEN**, **VON LINKS NACH RECHTS** usw. einstellen. Je nach deinen Wünschen für den Sound kannst du mit diesen Effekten noch viel Menge machen.



Für diesen Fall brauchen wir jedoch keinen Effekt. Unter SYNC gibst du an, wann Flash den Sound spielen soll. Folgende Eingaben sind möglich:

1. EREIGNIS

Flash spielt immer beim Erreichen dieses Schlüsselbildes den Sound ab. Auch wenn der Sound schon läuft!

Vorsicht: Das kann dann zu Überblendungen führen

2. START

Flash spielt diesen Sound beim Erreichen dieses Schlüsselbildes den Sound ab. Unterschied zum **Ereignis** ist, dass Flash den Sound nicht abspielt, wenn er schon läuft.

3. STOP

Mit Stop weist du Flash an, beim Erreichen dieses Schlüsselbildes alle Ereignis-Sounds und Stop-Sounds zu stoppen. Nur Streaming-Sounds laufen weiter.

4. STREAM

Sinnvoll für komplexe, langanhaltene Sounds wie z.B. bei richtigen Songs. Solche Sounds werden so lange abgespielt, wie die Ebene, auf der sie sich befinden, in der aktiven Zeitleiste verbleibt. Bei lang anhaltender Hintergrundmusiken ist dies eine sinnvolle Möglichkeit – Flash beginnt mit dem Abspielen, sobald

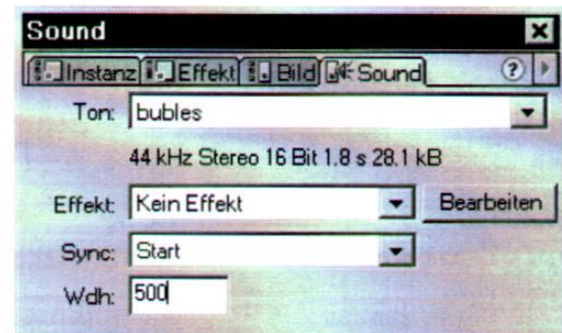
es einen kleinen Teil heruntergeladen hat.

Für unseren Sound wählst du hier START aus; denn die Datei selbst ist nur wenige Kilobyte groß und muss deshalb nicht gestreamt werden.

Unter WDH gibst du den Wert der Wiederholung des Sounds an. Soll er möglichst lange laufen, gibst du hier eine möglichst große Zahl ein.

Der maximale Wert für Wdh ist
2.147.483.647

Die Anzahl der Wiederholungen hat keinen besonderen Effekt auf die spätere Dateigröße und ist deshalb wirklich frei wählbar. Für dieses Spiel schlage ich 500 vor – und so sieht das Fenster dann aus.



Nun hast du passend zu deinen Luftblasen auch einen Sound.

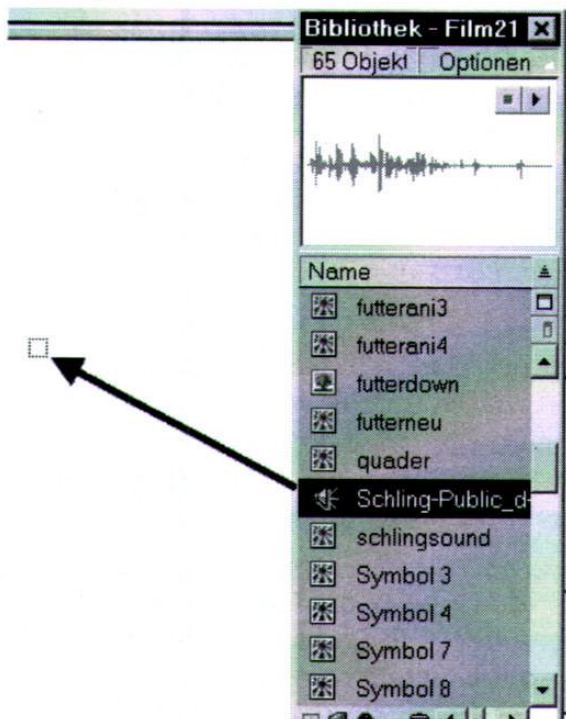
Fishbud macht beim Fressen Geräusche

Da du schon mal dran bist, kommt nun noch ein Sound für das Fressen und ein Sound für das GameOver hinzu. Für das Fressen von Fishbud habe ich die Datei **SCHLING.WAV** vorgesehen. Hierfür fügst du zuerst ein neues Symbol ein. Klick auf EINFÜGEN|NEUES SYMBOL. Als Name vergibst du *Schling*, als VERHALTEN lässt du FILMSEQEUNZ ausgewählt.

Da Flash sofort in den Überarbeitungsmodus für das neue Symbol springt, klickst du direkt mit der rechten Maustaste auf den Frame 1 und wählst AKTIONEN. Hier muss nun der `stop()`;-Befehl hinterlegt werden.

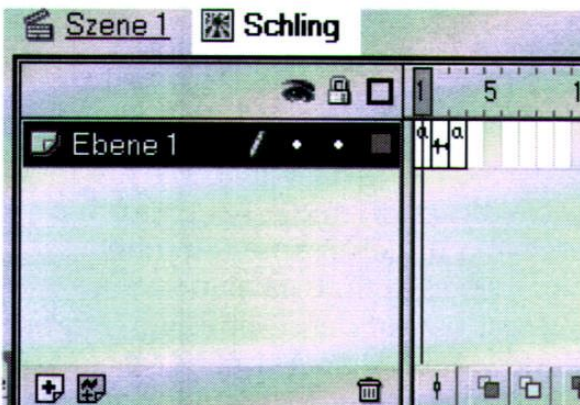
Dann klickst du mit der rechten Maustaste auf den Frame 2 und wählst LEERES SCHLÜSSELBILD EINFÜGEN. Nun

ziehst du per Drag & Drop aus dem Fenster BIBLIOTHEK die Sounddatei *Schling* auf die Bühne.



Anschließend fügst du in Frame 3 wiederum ein LEERES SCHLÜSSELBILD EIN und hinterlegst auch hier den ACTIONSCRIPT() Befehl `stop()` ;

Im Resultat sieht die Zeitleiste für das Symbol *Schling* so aus:



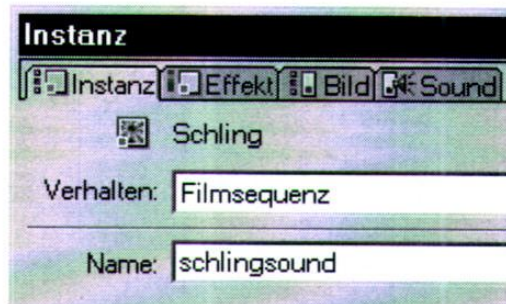
Da dieses Symbol später auf der Bühne hinterlegt wird, sind die `stop()` ;-Frames wichtig – schließlich soll der Sound nur dann abgespielt werden, wenn etwas gefressen wurde, und er soll in diesem Fall nur einmal abgespielt werden.

Per Klick auf SZENE 1 kehrst du in den Hauptfilm zurück. Hier klickst du die Ebene 1 an und ziehst dann aus dem Fenster BIBLIOTHEK das Symbol *Schling* irgendwo neben die Bühne.



Flash stellt dieses eigentlich leere Symbol als Kreis dar. Später im Film wird es nicht angezeigt, und deshalb könntest du dieses Symbol auch irgendwo *auf* die Bühne legen. Legst du es neben die Bühne, weißt du aber, wo es ist, und findest es schneller wieder.

Damit du später im Script besser auf dieses Symbol zugreifen kannst, solltest du nun noch einen Namen hierfür vergeben. Du klickst mit der rechten Maustaste auf den Sound und wählst `BEDIENFELDER|INSTANZ`. Hier gibst du unter `NAME` *schlingsound* ein.



Vergiss nicht, nach der Eingabe RETURN zu drücken. Flash neigt sonst dazu, die Eingabe zu übergehen!

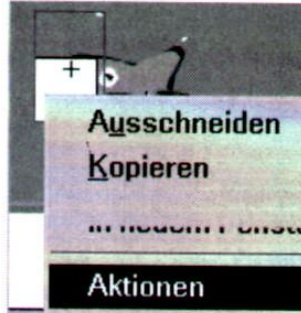
Nun zum Aufruf dieses Sounds. Er gehört in das Script, in dem du die Kollision mit den Futterportion prüfst. Sobald Flash die gefressene Futterportion löscht, muss dieser Sound aufgerufen werden.

Der Sound liegt auf `_level0` und hat den Namen `schlingsound`, d.h. du kannst ihn wie folgt aufrufen:

```
_level0.schlingsound.gotoAndPlay(2);
```

Bei dieser Zeile spielt Flash den Frame 2 dieser Filmsequenz ab. Da hier der Sound liegt und da du in Frame 3 ein `stop()`; eingebaut hast, stoppt Flash nach dem Sound wieder.

Du klickst doppelt auf Fishbud, um in den Bearbeitungsmodus zu gelangen. Hier wählst du mit den Frame 3 aus. In diesem Frame gibt es zwei Objekte: `fishbud`



und den `hitarea`. Um an das Script des `hitarea` heranzukommen, musst du einmal die Bühne anklicken und danach mit der rechten Maustaste den `hitarea`. Im erscheinendem Menu wählst du dann `Aktionen` aus.

Hier gibst du jeweils hinter dem Hochzählen der Variablen `gefressen` den vorher beschriebenen Aufruf für den `schlingsound` ein.

```
onClipEvent (enterFrame) {
    this._visible = false;
    if (this.hitTest(_root.futter0)) {
        removeMovieClip(_root.futter0);
        _root.gefressen++;
        _level0.schlingsound.gotoAndPlay(2);
        _level0.Fisch.gotoAndPlay(2);
    }
    if (this.hitTest(_root.futter1)) {
        removeMovieClip(_root.futter1);
        _root.gefressen++;
        _level0.schlingsound.gotoAndPlay(2);
        _level0.Fisch.gotoAndPlay(2);
    }
    if (this.hitTest(_root.futter2)) {
        removeMovieClip(_root.futter2);
        _root.gefressen++;
        _level0.schlingsound.gotoAndPlay(2);
        _level0.Fisch.gotoAndPlay(2);
    }
    if (this.hitTest(_root.futter3)) {
        removeMovieClip(_root.futter3);
        _root.gefressen++;
        _level0.schlingsound.gotoAndPlay(2);
        _level0.Fisch.gotoAndPlay(2);
    }
    if (this.hitTest(_root.futter4)) {
        removeMovieClip(_root.futter4);
        _root.gefressen++;
    }
}
```

Vergiss nicht, diese Zeile in allen acht Abfragen zu hinterlegen.

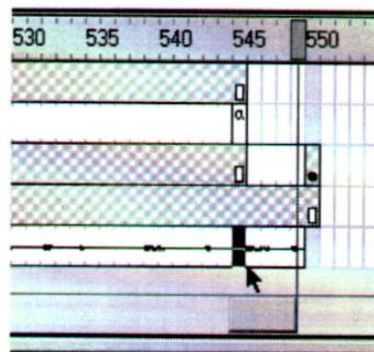
Achtung: Auch im `AKTIONSCRIPT()` des Frame 4, Fishbud frißt nach rechts schwimmend, musst du diese Erweiterung hinterlegen!

Hastdu alle diese Änderungen gemacht, macht Fishbud beim Fressen ein Geräusch.

GameOver Signal

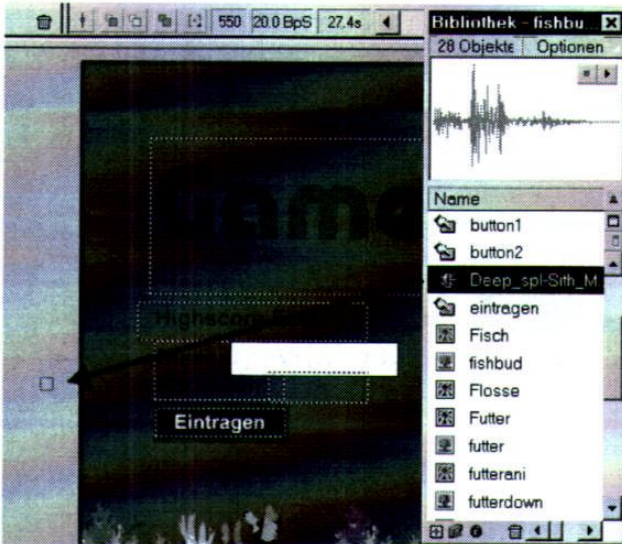
Fehlt noch das akustische Signal für das GameOver. Klicke auf SZENE 1, um in den Hauptfilm zurück zu kehren. In der untersten Ebene 5 liegt im Augenblick der Sound `bubbles`.

Scrolle nach rechts, bis du die Frames des GameOver-Bildes siehst. Der Tetriskbalken des Sounds reicht im Augenblick bis zum Frame 549. Zunächst ziehst du den Endframe des Sounds mit der Maus zurück auf den Frame 545.



Danach klickst du mit der rechten Maustaste auf den Frame 546 und wählst `LEERES SCHLÜSSELBILD EINFÜGEN`. Nun gehst du auf den GameOver-Frame 550 und wählst wiederum über die rechte Maustaste den Befehl `LEERES SCHLÜSSELBILD EINFÜGEN`.

Als Sound habe ich den Klang eines Steines gewählt, der ins Wasser geworfen wird. Dieser Sound heißt `DEPP_SPL_SITH_MAS-505.WAV` und ist von **Sith Master**. Diese Datei ziehst du per Drag & Drop aus dem Bibliothekfenster neben die Bühne.



Als Ergebnis siehst du, dass in der Zeitleiste im Frame 550 ein blaue Welle zu sehen ist. Wird dieser Frame nun aufgerufen, hört man den Steinwurf ins Wasser.

Soll das Blubbern auf dem GameOver-Frame aufhören, musst du noch ein paar Änderungen vornehmen. Zunächst verlängerst du die Tetrisbalken der Ebenen 1 und 4, indem du mit der Maus die Endframes dieser Ebenen auf den Frame 551 ziehst.

Dann klickst du den Soundframe 550 an und löschst ihn, entweder mit der ENTF-TASTE oder per BEARBEITEN| LÖSCHEN.

Anschließend klickst du den nun leeren Frame 550 mit der rechten Maustaste an und wählst LEERES SCHLÜSSELBILD EINFÜGEN; nachfolgend klickst du den Frame wiederum mit der rechten Maustaste an und wählst AKTIONEN. Hier gibst du nun den ACTIONSCRIPT()-Befehl `stopAllSounds()`; ein. Darunter kommt der Befehl `gotoAndStop(551);`.

Da der GameOver-Aufruf auch mit `gotoAndStop()`; erfolgen würde, würde ohne diesen Aufruf der Frame 551 nicht angezeigt!

Nun fügst du im leeren Frame 551 in der Sound-Ebene 5 ein LEERES SCHLÜSSELBILD EIN und ziehst wie vorher den Sound aus dem BIBLIOTHEK Fenster neben die Bühne. Fertig.

Durch `stopAllSounds()` werden alle laufenden Sounds gestoppt. Da der Steinwurfsound erst in Frame 551 erfolgt, ist er hiervon nicht betroffen. Fertig!

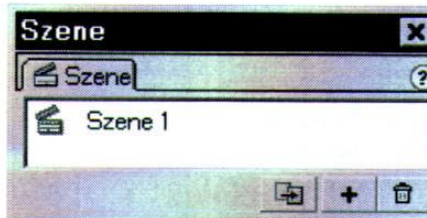
Nun hast du ein komplettes Spiel mit unterschiedlichen Schwierigkeitsgraden, einem animierten Spielfeld und Soundunterstützung. Diesen Stand kannst du als **FISHBUD8.FLA** auf www.spieleinflash.de herunterladen.

Der Preloader

Je nach Größe des Flashfilms kann so mancher Internetbesucher die Geduld verlieren, wenn er auf den Film- oder Spielstart wartet. Aus diesem Grund solltest du vor einem größeren Flashfilm oder auch Flashspiel einen Preloader einbauen, der z.B. über den Stand des Ladevorganges informiert.

Für den Standort des preloaders gibt es verschiedene Orte. So kannst du ihn z.B. vor dem Spiel in der Szene 1 platzieren. Hierbei müsstest du alle Tetrisbalken, die im Moment da sind, entsprechend nach hinten verschieben.

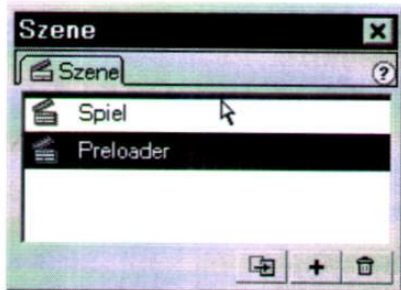
Eine weitere Möglichkeit ist das Einfügen einer weiteren Szene. Da du nun schon genug Tetrisbalken verschoben hast, mache ich mit dieser Variante weiter. Eine neue Szene fügst du per MODIFIZIEREN| SZENE ein. Es erscheint folgender Dialog:



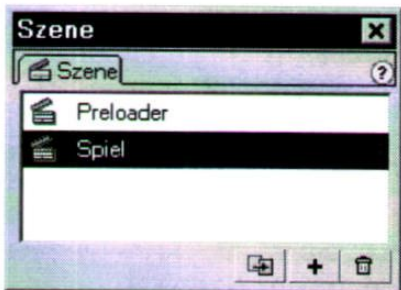
Zunächst gelangst du per Doppelklick auf den Titel *Szene 1* in den Umbenennungsmodus. Nenne diese Szene von nun an *Spiel*.

Bestätige den neuen Namen mit der Return-Taste. Flash neigt sonst zum Übergehen der Eingabe!

Nun fügst du mit einem Klick auf den **+**-BUTTON eine neue Szene ein. Diese *Szene 2* nennst du *Preloader*. Flash spielt die Szenen der Reihe nach ab. Im Augenblick kommt der Preloader also nach dem Spiel. Da er als erstes erscheinen soll, ziehst du ihn bei gedrückter Maustaste über das Spiel.



Nun sieht das SZENE Fenster so aus:



Klickst du nun den *Preloader* an, siehst du, dass dir eine neue, komplett leere Zeitleiste zur Verfügung steht. Klickst du das *Spiel* an, siehst du deine bekannte Spiel-Zeitleiste.

Über das SZENE Fenster kannst du also zwischen den einzelnen Szenen hin und her wechseln.

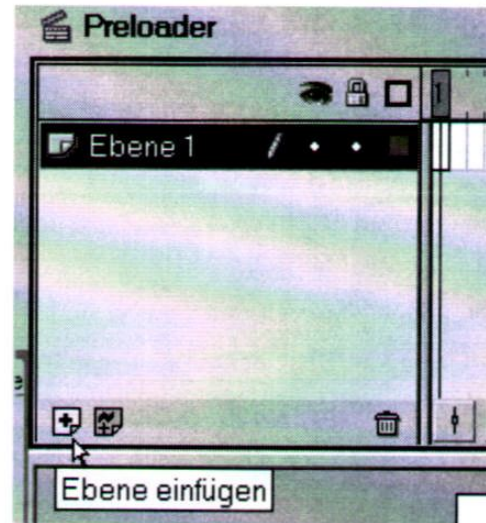
Nun klickst du den *Preloader* an. Das Fenster SZENE kannst du dann erst einmal schließen.

Zum Inhalt des preloaders: Zum einen muss geprüft werden, wie weit die Datei bereits geladen ist. Diesen Wert kannst du dann für eine Ausgabe am Bildschirm benutzen.

Neben diesem Ladestatus ist es üblich, dem Spieler auch eine kleine Anleitung zum Spiel zu geben. Sinnvoller Weise gehört diese Anleitung auch in den Preloader.

Nach dem der Film geladen ist, soll dann eine Schaltfläche vorhanden sein, die der Spieler anklickt, um zum Spiel

zu gelangen. Für diese Inhalte brauchst du drei Ebenen – also fügst du unmittelbar zwei weitere Ebenen ein, und zwar über den EBENE EINFÜGEN-BUTTON.



Die Ebene 1 machst du zu deiner Scriptebene. Du klickst mit der rechten Maustaste auf den Frame 1 in der Ebene 1 und wählst AKTIONEN.

Zunächst brauchst du eine Variable, in der du später den Wert des geladenen Status hinterlegen kannst. Nenn diese Variable **ausgabe** und gib ihr den Startwert 0.

```
var ausgabe();
```

Nun klickst du mit der rechten Maustaste den Frame 2 in der Ebene 1 an und wählst LEERES SCHLÜSSELBILD EINFÜGEN. Du gelangst in den ACTIONSCRIPT()-Modus, indem du wiederum mit der rechten Maustaste auf diesen Frame klickst und AKTIONEN wählst.

Hier kommen nun die Befehle hinein, die den Status des Ladevorganges prüfen und den hieraus resultierenden Wert in der Variablen **ausgabe** speichern.

Für die Prüfung des Ladestatus bietet Flash ein paar Hilfestellungen per ACTIONSCRIPT() an. Hierbei musst du unterscheiden, ob du den Spieler über die Anzahl der geladenen Bilder, also Frames, oder über den Wert der geladenen Bytes informieren willst.

1.) Info über die geladenen Bilder

Die Eigenschaften `_totalframes` und `_framesloaded` stehen dir hierfür zur Verfügung. Die Eigenschaft `_totalframes` beinhaltet den Wert der vorkommenden Einzelbilder, also Frames, deines Films. Die Eigenschaft `_framesloaded` beinhaltet die Zahl der bereits hiervon geladenen Bilder. Mit diesen zwei Werten kannst du z.B. folgende Meldung generieren:

"48 von 120 Bildern geladen."

Für eine solche Meldung änderst du den Wert der Variable `ausgabe` wie folgt:

```
_root.ausgabe=_framesloaded+"
Bilder von"+_totalframes+" Bilder
geladen";
```

Mit den `+`-ZEICHEN verbindest du hierbei starre Textpassage wie *Bilder von* mit Inhalten von Variablen wie `_framesloaded`.

Damit sich diese Anzeige auch dynamischer verhält, musst du nun eine Art Frame Schleife bauen, die immer wieder auf diesen Frame zurückkehrt.

So wird der Inhalt der Variablen `ausgabe` immer wieder aktualisiert. Hierzu klickst du den Frame 3 mit der rechten Maustaste an und wählst LEERES SCHLÜSSELBILD EINFÜGEN.

Dann klickst du den Frame erneut mit der rechten Maustaste an und wählst AKTIONEN. Als `ACTIONSCRIPT()` hinterlegst du hier den Aufruf des vorherigen Frames mit der Zeile `prevFrame()` ; .

Das ergibt eine Endlosschleife die, im Augenblick andauernd zwischen den Frames 2 und 3 läuft. Als Ergebnis erscheint eine sich ständig aktualisierende Variable `ausgabe`.

Damit du von diesem Effekt auch etwas hast, benötigst du noch ein Textfeld, das auf diese Variable zugreift. Hierfür klickst du in Ebene 2 mit der rechten Maustaste auf den Frame 2 und wählst LEERES SCHLÜSSELBILD EINFÜGEN. Dann nimmst du dir das TEXTWERKZEUG und malst ein Textfeld auf die Bühne. Klickst du dieses Textfeld mit der rechten

Maustaste an, kannst du per `BEDIENFELDER|TEXTOPTIONEN` die Zuweisung der Variable vornehmen.

Hierfür stellst du das Textfeld von STATISCHER TEXT auf DYNAMISCHER TEXT um. Unter dem Feld `VARIABLE` gibst du `ausgabe` ein. Im gleichen Fenster kannst du unter dem Reiter `ZEICHEN` noch `SCHRIFTART`, `SCHRIFTGRÖßE` und `SCHRIFTFARBE` einstellen – ein wenig Optik muss schließlich auch sein.

Du musst evtl. die Größe des Textfelders anpassen, damit der Text auch ganz angezeigt wird.

Die weitere Kosmetik des Preloaders ist die gleiche wie bei der Anzeige der geladenen Bytes. Bevor ich damit weitermache, erkläre ich nun kurz die zweite Variante der Statusanzeige.

2.) Info über die geladenen Bytes

Neben der Information, wie viele Bilder bzw. wie viele Bytes bereits geladen sind, sieht man häufig auch eine Prozentanzeige.

Für den Spieler ist diese Angabe erheblich aussagekräftiger, weswegen ich bei dieser Anzeige auch erkläre, wie du auf Basis der geladenen Bytes zu einer Prozentanzeige kommst. Dieses Verfahren basiert auf einem einfachen Dreisatz, nämlich

Geladenes geteilt durch das Gesamte multipliziert mit 100.

Das kannst du entsprechend natürlich auch bei der Anzeige auf Basis der einzelnen Bilder machen.

Ähnlich wie bei den Bildern bietet Flash für den Vergleich der geladenen Bytes und der Gesamtbytezahl zwei Eigenschaften an: `getBytesLoaded()` und `getBytesTotal()`

Wie vorher könntest du einfach eine Anzeige bauen wie

"28 Bytes von 64 Bytes geladen"

Die `ACTIONSCRIPT()` Zeile sähe dann so aus:

```
ausgabe=getBytesLoaded()+" Bytes
von" + getBytesTotal()+" Bytes
geladen";
```

Für die Anzeige der Prozentzahl musst du hier eine zusätzliche Zeile einfügen. In dieser Zeile generierst du eine Variable mit dem Namen `prozent`; als Inhalt soll die Dreisatzrechnung von oben hinein. In `ACTIONSCRIPT()` sieht das dann so aus:

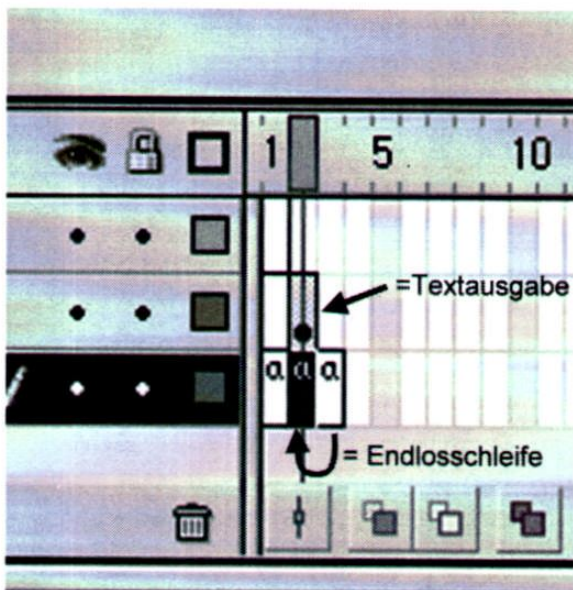
```
prozent=
Math.round(getBytesLoaded()/getByt
estotal()*100);
```

Die Funktion `Math.round()`; solltest du hierbei auf jeden Fall verwenden. So verhinderst du das Auftauchen von krummen Zahlen in deiner Anzeige.

Für die Füllung der Variable `ausgabe` kommt dann folgende `ACTIONSCRIPT()` Zeile:

```
ausgabe=prozent+" Prozent
von"+getBytesTotal()+" Bytes
geladen!";
```

Für die Ausgabe der Variablen `ausgabe` verwendest du ebenso ein Textfenster wie bei der Anzeige der geladenen Bilder. Im Resultat sieht deine *Preloader* Zeitleiste so aus:



Nun benötigt das Script, in dem du die Variable `ausgabe` mit neuen Inhalten füllst, eine Abfrage, die aus der Endlosschleife aussteigt, wenn 100% geladen sind. Diese Abfrage kannst du mit der

If-Abfrage machen. Klar formuliert heißt die Abfrage:

```
Wenn (geladene Bytes = totale
Bytes) dann { springe zum Frame 5}
```

Da Frame 5 ausserhalb der Endlosschleife liegt, springst du durch diesen Aufruf aus der Endlosschleife heraus.

In `ACTIONSCRIPT()` sieht die Schleife dann so aus:

```
if
(getBytesLoaded()==getBytesTotal()
) {gotoAndPlay(5);}
```

Diese Zeile schiebst du vor die Generierung der Variablen `prozent`, so dass das Script dann so aussehen sollte:

```
if
(getBytesLoaded()==getBytesTotal()
) {gotoAndStop(5);}
prozent=Math.round(getBytesLoaded(
)/getBytesTotal()*100);
```

```
_root.ausgabe=prozent+" Prozent
von "+getBytesTotal()+" geladen!";
```

Als nächstes soll in Frame 5 ein weiterer Inhalt erscheinen. Ich schlage vor, dass du zum einen den momentan sichtbaren Text langsam verschwinden lässt und gleichzeitig Fishbud an dieser Stelle auftaucht.

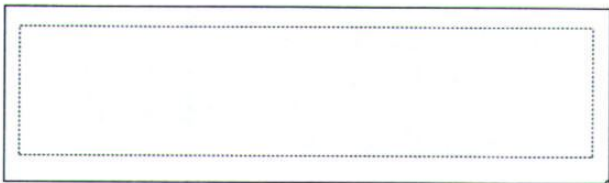
Unter Fishbud setzt du dann noch einen Text zur Spielerklärung und die Aufforderung, Fishbud anzuklicken, damit das Spiel losgeht. Im Einzelnen gehst du wie folgt vor:

Da Textfenster mit dynamischem Inhalt nicht als Symbol verwendet werden können, musst du nun ein wenig improvisieren!

Du klickst mit der rechten Maustaste in Ebene 3 auf den Frame 5. Nun wählst du das **RECHTECKWERKZEUG** aus und stellst die Farben für den Rahmen und die Füllung auf weiß.



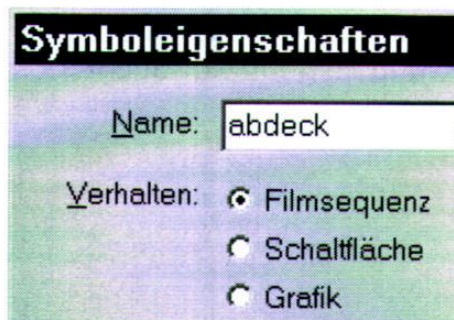
Mit diesem Werkzeug malst du ein Rechteck über dein Textfenster und deckst es so komplett ab.



Da du die Farben weiß gewählt hast, ist im Moment nichts mehr zu sehen. Du gehst nun mit dem Pfeilwerkzeug auf das unsichtbare Rechteck und machst einen Doppelklick, um es zu markieren.



Nun wählst du im Menü MODIFIZIEREN|GRUPPIEREN aus – und dann im Menü EINFÜGEN|IN SYMBOL KONVERTIEREN. Als NAME vergibst du *abdeck*, als VERHALTEN lässt du FILMSEQUENZ eingestellt.



Dann klickst du den Frame 15 in der Ebene 3 mit der rechten Maustaste an und wählst SCHLÜSSELBILD EINFÜGEN.

Dieses Abdeckrechteck soll sich nun von unsichtbar in sichtbar wandeln. Hierfür gibt es den EFFEKT ALPHA. Hat das Symbol den Alphawert 0 ist es unsichtbar, hat es den Wert 100, ist es voll sichtbar. Das Symbol **abdeck** muss also in Frame 5 den Alphawert 0 bekommen.

Hierfür klickst du den Frame 5 in Ebene 3 an. Dann klickst du mit der rechten Maustaste auf das gemalte Rechteck und wählst im Menü BEDIENFELDER|EFFEKT aus.

Hier stellst du dann ALPHA ein und wählst als Wert 0 aus.



Dann klickst du mit der rechten Maustaste auf den grauen Tetriskbalken zwischen Frame 5 und Frame 15 und wählst BEWEGUNGSTWEEN EINFÜGEN.

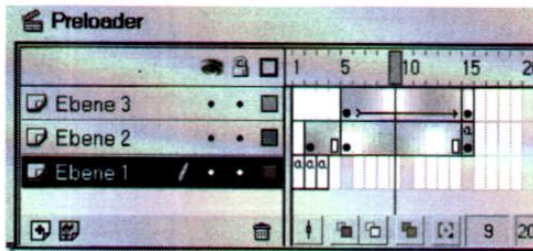
Im Resultat verwandelt sich der graue Tetriskbalken in einen lilafarbenen mit einem schwarzen Pfeil. Später im Flashfilm überdeckt das langsam auftauchende, weiße Rechteck den Ladestatus.

Nun verlängerst du den Tetriskbalken in Ebene 2 indem du mit der rechten Maustaste in Ebene 2 auf den Frame 15 klickst und SCHLÜSSELBILD EINFÜGEN wählst.

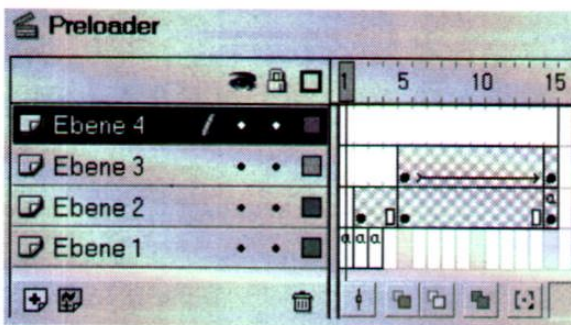
Willst du diesen Stand schon mal testen, musst du in diesem Schlüsselbild noch den ACTIONSCRIPT() Befehl `stop()`; einbauen. Andernfalls spielt Flash nach dieser Szene sofort die nächste Szene ab und startet damit automatisch das Spiel.

Hierfür klickst du den Frame 15 in Ebene 2 mit der rechten Maustaste an und wählst AKTIONEN.

In diesem Menü kannst du dann den **stop()** ;-Befehl selber eingeben oder ihn aus der Liste am linken Rand per Doppelklick einfügen. Im Resultat sollte nun deine Zeitleiste so aussehen:



Nun kommt unser Held Fishbud hinzu. Hierfür fügst du zuerst eine weitere Ebene ein. Du klickst die Ebene 3 an und klickst auf den Button EBENE EINFÜGEN. Diese neue Ebene 4 legt sich automatisch über die Ebene 3, und alle Ebenen liegen nun in der richtigen Reihenfolge.



Beim Fishbud ist es wichtig, dass er auch ein Symbol ist. Genau wie beim überdeckenden Rechteck stellst du den ALPHAWERT beim ersten Schlüsselbild auf 0.

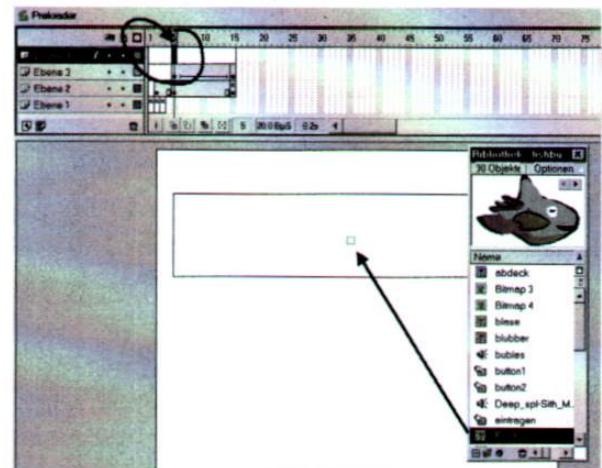
Zunächst klickst du mit der rechten Maustaste den Frame 5 in Ebene 4 an und wählst SCHLÜSELBILD EINFÜGEN. Da Fishbud ja bereits als Symbol vorhanden ist, kannst du ihn dir aus dem Fenster BIBLIOTHEK herein ziehen. Sollte das Fenster nicht geöffnet sein, öffnest du es per Menü FENSTER|BIBLIOTHEK.

Im Bibliothek-Fenster scrollst du, bis du Fish siehst.

Fishbud ist die importierte GIF Grafik und *Fisch* die von dir in eine Vektorgrafik umgewandelte und animierte Flashgrafik!

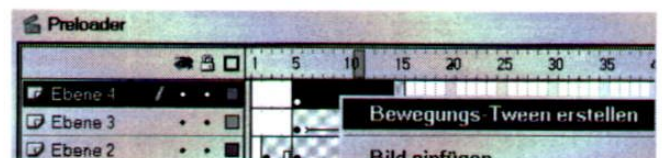
Nun kannst du *Fisch* per Drag & Drop mit der Maus auf die Bühne ziehen.

Achte darauf, dass der Frame 5 in Ebene 4 schwarz markiert ist. Flash fügt die Symbole immer in den aktiven Frame ein.



Am besten positionierst du *Fisch* in der Mitte des Textfensters. Dann klickst du mit der rechten Maustaste auf den Frame 15 und wählst SCHLÜSELBILD EINFÜGEN. Nun klickst du wieder den Frame 5 in Ebene 4 an.

Hier klickst du *Fisch* mit der rechten Maustaste an und wählst **BEDIENFELDER|INSTANZ**, dann unter dem Reiter **EFFEKT** den Punkt **ALPHA**, und stellst als Wert 0 ein. Anschließend klickst du den grauen Tetrusbalken zwischen 5 und 15 mit der rechten Maustaste an und wählst **BEWEGUNGSTWEEN ERSTELLEN**.



Als Ergebnis hast du in Ebene 3 und Ebene 4 zwei lilafarbene Tetrusbalken mit einem schwarzen Pfeil. Im Preloader siehst du, wie der Ladestatus langsam verschwindet und Fishbud auftaucht.

Spielanleitung

Zunächst klickst du mit der rechten Maustaste in Ebene 4 auf den Frame 16 und wählst **SCHLÜSSELBILD EINFÜGEN**. Danach klickst du mit der rechten Maustaste in der Ebene 3 auch den Frame 16 an und wählst **Leeres Schlüsselbild** einfügen. In dieses leere Schlüsselbild kommt nun

der Text für die Spielanleitung. Da wir gerade schon mit einer Animation gearbeitet haben, sollten die weiteren Texte auch mit einem Effekt auf den Bildschirm kommen. Ich schlage vor, dass die Texte von links und rechts auf die Bühne fliegen sollen. Hierfür schreibst du mit dem **TEXTWERKZEUG** erst einmal den Text für die Spielbeschreibung, z.B. so:



Bevor es los geht, gibt es hier noch eine kurze Spielanleitung:

Du steuerst Fishbud mit dem Cursor-Block auf deiner Tastatur! Die Schwimmrichtung entspricht dabei den Pfeilsymbolen!

Um zu fressen, betätigst du die Leertaste!

Das Spiel ist vorbei, wenn du mehr als 5 Portionen verpasst hast!

Ist der Text fertig, wählst du **EINFÜGEN|IN SYMBOL KONVERTIEREN**. Als **NAME** vergibst du *anleitung*, als **VERHALTEN** lässt du **FILMSEQUENZ** eingestellt. Nun fügst du in Ebene 2 auf Frame 16 auch ein **Leeres Schlüsselbild** ein.

Da der Ladestatus hier nicht mehr wichtig ist, kannst du nun die Ebene 2 für etwas anders nutzen.

Du nutzt das Textwerkzeug und fügst hier nun einen solchen Text ein:

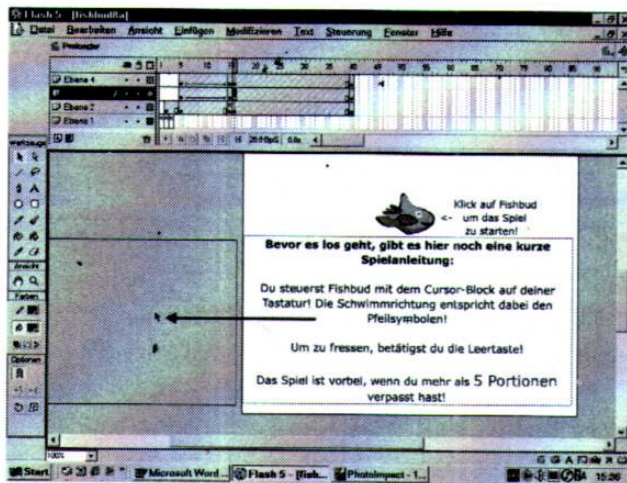


Klick auf Fishbud
<- um das Spiel
zu starten!

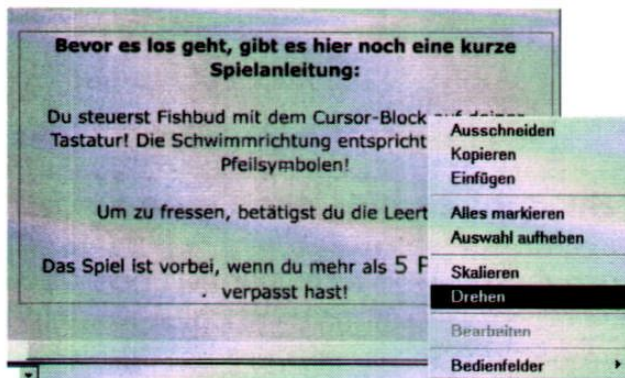
geht, gibt es hier noch eine kurze Spielanleitung:

Auch diesen Text wandelst du über **EINFÜGEN|IN SYMBOL KONVERTIEREN** in ein Symbol um. Als **NAME** vergibst du hier *starttext*, als **VERHALTEN** lässt du auch hier **FILMSEQUENZ** aktiviert. Damit die Texte auf die Bühne fliegen, musst du zunächst in beiden Ebenen ein weiteres Schlüsselbild einfügen, z.B. bei Frame 40. Hierfür klickst du nacheinander mit der rechten Maustaste auf die Frames 40 in den Ebenen 2, 3 und 4 und wählst **SCHLÜSSELBILD EINFÜGEN**.

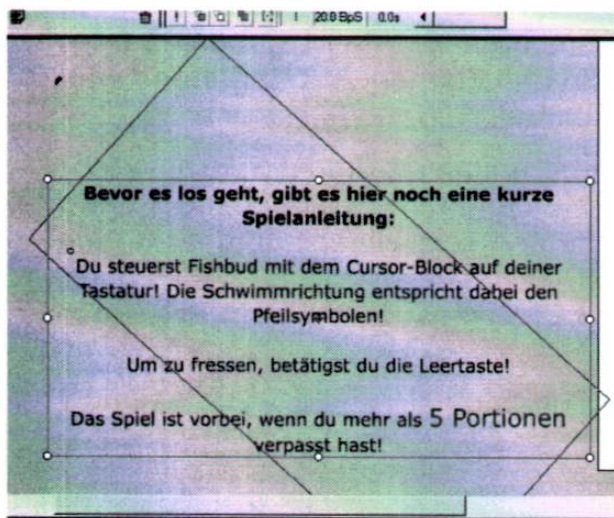
Zuerst animierst du die Spielanleitung. Wähle den Frame 16 in Ebene 3 und ziehe den Text einfach von der Bühne nach links.



Da du den Text ohnehin gepackt hast, kannst du einen weiteren Effekt hinein bringen – du kannst ihn z.B. um 270 Grad drehen, und später in der Animation dreht sich der Text dann auf die Bühne! Hierfür klickst du den Text mit der rechten Maustaste an und wählst **DREHEN**.



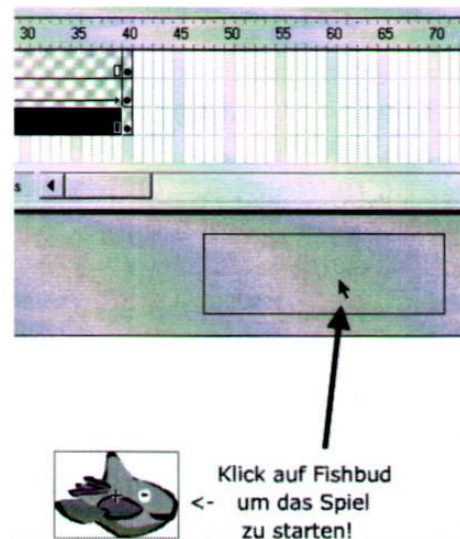
Das Ergebnis zeigt an den Eckpunkten kleine weiße Kreise. Klickst du einen dieser Kreise an und hältst die Maustaste gedrückt, kannst du den Text drehen.



Achte darauf, dass du den Text nicht zufällig doppelt anklickst – sonst gelangst du in den Überarbeitungsmodus für das Symbol und das Drehen passiert für dieses Symbol nicht nur im Schlüsselbild 16 sondern allgemein gültig!

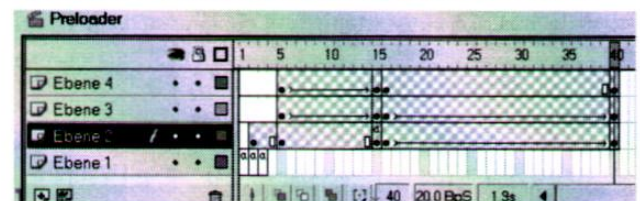
Nun klickst du mit der rechten Maustaste auf den Tetrisbalken zwischen Frame 16 und Frame 40 und wählst BEWEGUNGSTWEEN ERSTELLEN. Als Ergebnis generiert Flash eine Animation des von links auf die Bühne rollenden Textes. Entsprechend gehst du nun mit der Klickaufforderung vor, d.h. du wählst in der Ebene 2 den Frame 16 an. Dann nimmst du den Text und ziehst ihn nach oben von der Bühne.

Auch hier kannst du mit dem Text noch ein wenig herum experimentieren, z.B. eine Drehung und einen veränderten Alphawert einfügen.



Für die Drehung klickst du den Text mit der rechten Maustaste an und wählst DREHEN. Hast du den Text über die Eckkreise gedreht, klickst du ihn erneut mit der rechten Maustaste an und wählst BEDIENFELDER]

EFFEKT, hier dann den Effekt ALPHA, und stellst als Wert 0 ein. Zum Schluß klickst du den grauen Tetrisbalken zwischen Frame 16 und Frame 40 mit der rechten Maustaste an und wählst BEWEGUNGSTWEEN ERSTELLEN. Im Resultat sieht deine Zeitleiste dann so aus:



Damit diese Animation auch läuft, musst du nun noch den `stop()`;-Befehl in Ebene 2 in Frame 15 herausnehmen. Hierfür klickst du diesen Frame mit der rechten Maustaste an und wählst AKTIONEN aus. In diesem Fenster markierst du den `stop()`;-Befehl mit der Maus und löschst ihn mit der ENFTASTE .

Dieser `stop()`;-Befehl gehört nun in den Frame 40, am besten in die Ebene 4. Hierfür klickst du mit der rechten Maustaste in Ebene 4 auf den Frame 40 und wählst AKTIONEN. Entweder wählst du den EXPERTENMODUS aus und tippst den `stop()`;-Befehl manuell ein oder fügst ihn per Doppelklick aus der Liste ein.

Spielstart

Fehlt nur noch die Abfrage für Fishbud in Frame 40. Wird er angeklickt, soll das Spiel los gehen.

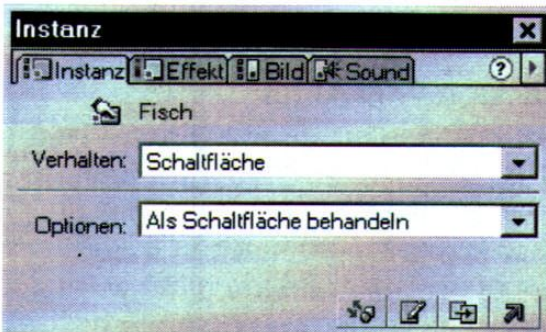
Also klickst du Fishbud in Frame 40 mit der rechten Maustaste an und wählst AKTIONEN. Hier muss nun ein Script mir folgender Logik hinterlegt werden:

Wenn (die Maustaste losgelassen wird) dann {rufe den ersten Frame in der Szene Spiel auf}

In ActionScript() sieht dies so aus:

```
on (release) {  
gotoAndPlay("Spiel", "1");  
}
```

Damit dieses Script funktioniert, musst du noch das Verhalten von *Fishbud* für dieses Schlüsselfeld ändern, und zwar in SCHALTFLÄCHE. Du klickst Fishbud mit der rechten Maustaste an und wählst **BEDIENFELDER|INSTANZ**. In diesem Fenster kannst du das VERHALTEN des Symbols einstellen. Wähle hier **SCHALTFLÄCHE**.



Testest du nun das Spiel, funktioniert im letzten Frame das Symbol *Fisch* wie eine Schaltfläche und startet beim Mausklick das Spiel!

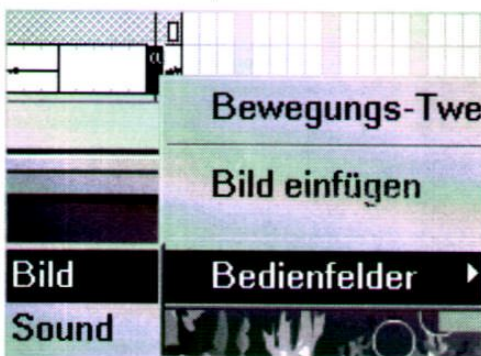
Frameaufruf bei mehreren Szenen

Später, wenn der Flashfilm veröffentlicht ist, gibt es für den Flash-Player keine einzelnen Szenen mehr. Du musst von nun an also aufpassen, welche Frames du mit dem ACTIONSCRIPT()-Befehl `gotoAndStop()` bzw.

`gotoAndPlay()` aufrufst. Innerhalb einer Filmsequenz ist das nicht sonderlich schlimm; z.B. beim Aufruf des GameOver-Fensters gibt es von nun aber ein Problem. Hier rufst du nämlich den Frame 550 auf `_level0` auf. Da Flash die 40 Frames des Preloaders vor die 550 Frames des Spiels setzt, musst du hier also nicht den Frame 550 aufrufen, sondern den Frame 590!

Im allgemeinen beginnt man hier auch mit der Vergabe von Framenamen.

Willst du dem GameOver-Frame einen Namen geben, gehst du wie folgt vor: Du wählst die Szene *Spiel* aus. Hier scrollst du vor bis zum Frame 550. Du klickst mit der rechten Maustaste den Frame 550 in Ebene 5 an und wählst **BEDIENTFELDER|BILD**.



In diesem Menü kannst du dem Frame unter **BEZEICHNUNG** einen Namen geben. Gib z.B. *gameover* als **BEZEICHNUNG** ein.

Bestätige die Eingabe mit Return, da Flash sonst dazu neigt, die Eingabe zu übergehen!

Nun scrollst du in der Zeitleiste bis zum Frame 1 zurück und klickst oben in der Zeitleiste diesen ersten Frame an. Du siehst unter anderem auch dein Originalfutter. Klickst du dieses doppelt an, gelangst du in den Bearbeitungsmodus für *futterani*. Hier scrollst du bis zum Frame 160 und klickst diesen Frame mit der rechten Maustaste an. Dann wählst du im Menü **AKTIONEN** – und gelangst in den ACTIONSCRIPT()-Modus, wo du den Aufruf des Frames 550 in den Aufruf des Frames *gameover* ändern kannst.

Das ActionScript() sieht dann so aus:

```
stop ();
_root.verloren++;
if (_root.verloren>5) {
_level0.gotoAndStop("gameover");
}
```

Fertig!

Das Spiel Fishbud ist nun komplett. Diesen Stand kannst du dir auf www.spieleinflash.de herunterladen. Die Datei heißt **FISHBUD9.FLA**.

Schlusswort vom Autor

Ich hoffe sehr, dass du mit diesem Heft durch alle komplexen Strukturen eines Flash-Spiels hindurch kommst. Um die Orientierung in einem solchen mehrdimensionalen Daumenkino nicht zu verlieren, geht man am besten Schritt für Schritt vor. Deshalb habe ich den Aufbau des Spieles genau so beschrieben, wie man ihn "Schritt für Schritt" machen würde.

Solltest du Fragen zum Thema oder zum Heft haben, dann schreib mir doch einfach eine E-Mail oder schau auf der Webseite www.spieleinflash.de vorbei. Wenn du ein Thema vermisst hast, dann teile es mir bitte auch mit.

Evtl. kann dein Thema dann in einem späterem Heft verarbeitet werden.

Ich wünsche dir viel Spaß beim Programmieren mit "Spiele in Flash".

oliverlange@knowware.de

- _framesloaded, 61
- _root, 12
- _totalframes, 61
- Animieren, 7
- Anzahl der Portionen erhöhen, 48
- Anzeigedauer verlängern, 47
- Asynchrone Bewegung, 52
- Bewegungs-Tween erstellen, 16
- Bildrate, 6
- Bps, 15
- Button erstellen, 30
- Do-While-Schleife, 19
- Drehbuch, 5
- duplicateMovieClip, 20
- Ellipsenwerkzeug, 53
- Farbe füllen, 28
- Filmeigenschaften, 6
- Flash Animation, 15
- For-Schleife, 19
- Framenamen, 68
- Frames verschieben, 18
- getBytesLoaded(), 61
- getBytesTotal(), 61
- getURL(), 32
- gotoAndPlay(), 25
- Grafiken importieren, 15
- Grafiken Vektorisieren, 7
- highscore.txt, 41
- Hintergrundgrafik, 27
- hitTest, 21
- HTML und PHP, 32
- If-Schleife, 26
- Instanznamen, 8; 18
- Key.getCode(), 9
- Kollisionskontrolle, 21
- Leeres Schlüsselbild einfügen, 30
- Level, 29
- Levelanzeige, 49
- loadVariablesNum(), 41
- lokale Variable, 51
- Math.round(), 62
- Mausklick Aktion, 32
- Maussteuerung, 14
- Modifizieren
 - Gruppieren, 7
- Mouse.hide(), 14
- MovieClip**, 6
- Objektaktionen, 9
- Objekte
 - (un-)sichtbar machen, 20
 - auf der Zeitleiste arrangieren, 17
- Objekte auf einem Pfad verankern, 53
- on(release), 32
- onClipEvent(enterFrame), 21
- onClipEvent(keyDown), 9
- onClipEvent(load), 14
- Pfad, 53
- Pfadbene einfügen, 52
- Preloader, 59
- Prozentanzeige, 61
- Punkttestand, 24
- Randbegrenzung, 44
- removeMovieClip(), 22
- Schaltfläche erstellen, 30
- Schlangenlinie
 - Sinusfunktion, 50
- Schlüsselbild einfügen, 6
- Sound
 - Einblenden, Ausblenden, Von links nach rechts, 55
 - Spiellänge verändern, 55
 - Sync, 56
- Sound integrieren, 55
- Symbol, 6
- Symbole positionieren, 16
- Synchrone Bewegung, 52
- Szene
 - Frameaufruf, 68
- Szene aufrufen, 67
- Szene einfügen, 59
- Szene wechseln, 60
- Tastaturcode, 9
- Tastatursteuerung, 8
- Teile einer Grafik animieren, 7
- Text drehen, 65
- Text verschwinden lassen, 62
- Textfeld als Punkteanzeige, 24
- Textfeld einfügen, 39
- Textfenster animieren, 65
- this.StartDrag(true), 14
- Transformieren
 - Drehen, 11
 - Horizontal Spiegeln, 10
 - Skalieren, 7
- Variable, 12
- Variablen
 - Laden aus PHP, 39
 - Übergabe an PHP-HTML, 33
- Verankern, 53
- Wenn-Dann-Sonst-Formel, 13
- Zeichen, 24
- Zeichnen
 - Rechteck, 27
- Zufallszahl generieren, 19

Preis: 4,- EUR pro Heft

Ausverkauft, dann kostenlos als PDF-Datei auf der Homepage

Bonner Presse Vertrieb macht für KnowWare Bestellungen: online via www.knowware.de oder per Telefon, Fax oder Brief:

Bonner Pressevertrieb, Moeserstr. 2-3,
49074 Osnabrück, knowware@bpv-online.com
Tel: +49 (0)541 33145-20
Fax: +49 (0)541 33145-33

Versand und Verpackung in EUR - Deutschland
2 bis 3 Hefte - 2,60 bis 7 Hefte - 5 bis 10 Hefte
und darüber: kostenfrei.

Auslandsporto: siehe www.knowware.de

St	Programmierung	
	Batchprogrammierung DOS	125
	C++ für Einsteiger	E06
	CGI & Perl für Einsteiger	P15
	Java2 für Einsteiger	P19
	Visual Basic für Einsteiger	S04*
	Windows	
	Start mit Windows 3.11	105
	Start mit Windows 95	139
	Windows 95 für Einsteiger	148
	Windows 98 für Einsteiger	158
	Windows 2000 für Einsteiger	E05
	Windows 2000 für Fortg.	P17
	Windows ME/98 für Einst.	166
	Windows-Netzwerke Einst.	E14*
	Windows Tips und Tricks	P02
	Windows-Tuning - Registry	P01
	Windows XP für Einsteiger	P22
	Windows Super User	P25
	Word	
	Word 7 für Anfänger	129
	Word 97 für Anfänger	E03*
	Word 7 für Fortgeschrittene	132
	Weiter mit Word 97/2000	160
	Word f Studenten 7/97/2000	138
	Word 2000 für Einsteiger	164*
	Word 2002 für Einsteiger	171
	Xtra - Diverse	
	Acrobat und PDF für Einst.	E10
	ISDN für Einsteiger	P13
	Rund um den PC	143
	Staroffice 5.x für Einsteiger	P09
	Viren, Hacker, Firewalls	170
	Was ist denn DOS?	104
	Office 2000 Sekretäre/innen	S06

Name, Anschrift, E-Mail und Tel:

St	Datenbank: Access, SQL	
	Start mit Access 2	107
	Start mit Access 7/97	146*
	Access 2000 für Einsteiger	162*
	Access 97/2000 für Fortg.	154*
	Access 2002 für Einsteiger	172
	Access: Formulare und Berichte	P18
	Start mit Datenbanken und SQL	131*
	Excel	
	Excel 7 für Anfänger	135
	Start mit Excel (7, auch 5/97)	145
	Weiter mit Excel (Ver. 5/7)	112
	Excel VBA Makro-Programmier.	126
	Excel 97 für Einsteiger	156*
	Excel 97 für Fortgeschrittene	155*
	Excel 2000 für Einsteiger	E02*
	Excel 2000 für Fortg.	P20
	Grafik	
	Bildbearbeitung für Einsteiger	P16
	CorelDraw 7-10 für Einsteiger	P23
	Paint Shop Pro 5/6 für Einsteiger	P10
	PhotoShop LE für Einsteiger	S05
	PhotoShop 6.0 für Einsteiger	E15
	Hardware	
	CD-Brennen für Einsteiger	S02
	CD-Brennen mit Nero 5.5	P26
	Musik bearbeiten am PC	E11
	Homepages	
	Barrierefreies Webdesign	E08
	Dreamweaver 3/4 für Einsteiger	P14
	Flash5 für Einsteiger	E09
	Frontpage 2000 für Einsteiger	159*
	GoLive für Einsteiger	P21
	HomePages für Einsteiger	161*
	WWW - Homepages selbst erstellen	122*
	HomePages mit HTML und CSS	168*
	HomePages für Fortg.	P12*
	Intranet, HTML und Java	133
	JavaScript für Einsteiger	P06*
	JavaScript für Fortgeschrit	P24
	PHP für Einsteiger	E12
	PHP und MySQL Einsteiger	E07*
	XML für Einsteiger	E13
	Internet	
	Start ins Internet, 4. Ausg.	157*
	Internet für Einsteiger, 3. Ausg.	173*
	Internet Explorer 4 für Einsteiger	152
	E-Mail mit Outlook Express 5/6	P08*
	Outlook 98/2000 für Einsteiger	S03*
	Outlook 98/2000/2002 Einst.	165*
	Linux	
	Linux für Einsteiger	153
	Linux im Netzwerk	P11
	PowerPoint	
	Start mit PowerPoint 7	140*
	PowerPoint 2000 für Einsteiger	S01*

Bestseller im KnowWare Verlag

KnowWare
Beispiele und Übungen
88 Seiten

**Windows ME/98
für Einsteiger**
Kursmaterial



www.KnowWare.de Johann-Christian Hanke

ISBN 3-89-54-178-1/4 - LIT 11.000 DM B.

KnowWare
Übungen und Beispiele
80 Seiten

**Internet
für Einsteiger**




www.KnowWare.de Johann-Christian Hanke

KnowWare EXTRA
Peer-to-Peer-Netze mit 95/98/2000/ME

**Windows-Netzwerke
für Einsteiger**

3. Ausgabe - 72 Seiten



Johann-Christian Hanke
www.KnowWare.de

Deutschland 4,- EUR Österreich 4,90 EUR
Schweiz 5,95 Ffr. Luxemburg 4,70 EUR Italien 5,50 EUR

KnowWare
So geht's Deiner Schritt für Schritt
3. Ausgabe

**HomePages
für Einsteiger**


1. vollständige Ausgabe
75 Seiten



www.KnowWare.de Johann-Christian Hanke

KnowWare
Übungen und Beispiele
88 Seiten

**Homepages
mit HTML und CSS**



www.KnowWare.de Johann-Christian Hanke

Deutschland 4,- EUR Österreich 4,90 EUR
Schweiz 5,95 Ffr. Luxemburg 4,70 EUR Italien 5,50 EUR

KnowWare PLUS
JavaScript, XHTML, DHTML und CSS

**HomePages
für Fortgeschrittene**



80 Seiten, davon 12 Seiten HTML/CSS-Referenz

www.KnowWare.de Johann-Christian Hanke

KnowWare PLUS
Jetzt 72 Seiten - Praxis und Fun

**JavaScript
für Einsteiger**



Martin Boier

www.KnowWare.de 2. Ausgabe

KnowWare EXTRA

**PHP & MySQL
dynamische Webseiten**

Zugriff auf Datenbankanhalte



Petra Bilke
www.KnowWare.de

Deutschland 4,- EUR Österreich 4,90 EUR
Schweiz 5,95 Ffr. Luxemburg 4,70 EUR Italien 5,50 EUR

KnowWare
Übungen und Erläuterungen
Neuauflage

**Excel 97
für Einsteiger**



www.KnowWare.de Palle Granbæk

KnowWare
Übungen und Beispiele

**Access 2000
für Einsteiger**



www.KnowWare.de Koore Thomsen/Pia Hardy

Deutschland 4,- EUR Österreich 4,90 EUR
Schweiz 5,95 Ffr. Luxemburg 4,70 EUR Italien 5,50 EUR

KnowWare EXTRA

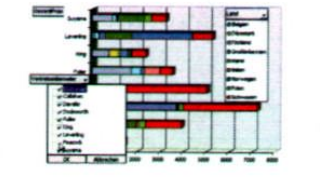
**Excel 2000
für Einsteiger**



Palle Granbæk
www.KnowWare.de

KnowWare PLUS
Beispiele und Übungen

**Excel 2000
für Fortgeschrittene**



www.KnowWare.de Thomas Barkow

Deutschland 4,- EUR Österreich 4,90 EUR
Schweiz 5,95 Ffr. Luxemburg 4,70 EUR Italien 5,50 EUR

KnowWare
Anleitungen und Workshops
2. aktualisierte Ausgabe, jetzt 80 Seiten

**Word 2000
für Einsteiger**



www.KnowWare.de Johann-Christian Hanke

KnowWare SPECIAL

**PowerPoint 2000
für Einsteiger**



www.KnowWare.de Johann-Christian Hanke

KnowWare PLUS
E-Mail ohne Probleme

**E-Mail mit
Outlook Express 5/6**

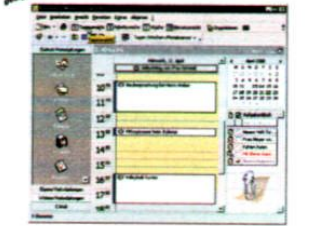


www.KnowWare.de Johann-Christian Hanke

Deutschland 4,- EUR Österreich 4,90 EUR
Schweiz 5,95 Ffr. Luxemburg 4,70 EUR Italien 5,50 EUR

KnowWare
Anleitungen und Workshops
2. aktualisierte Ausgabe, jetzt 80 Seiten
98/2000/2002

**Outlook
für Einsteiger**



www.KnowWare.de Johann-Christian Hanke