

# Workshop: Director FOLGE 9

## Objekt-Spielereien

Mit dem Spiel „Snake“ geben wir Ihnen einen Einblick in die objektorientierte Director-Programmierung.



■ In dieser Folge unseres Workshops wollen wir ein Spiel namens „Snake“ realisieren (Abbildung 1). Die Spielidee ist simpel: Eine anfangs kleine Schlange wird mit den Pfeiltasten über das Spielfeld bewegt und „frisst“ andere Objekte, wo immer sie auf diese trifft. Mit jeder Mahlzeit wächst die Schlange ein Stück – was schließlich zum Problem wird, da sie bei der Bewegung auf dem Spielfeld weder den Spielfeldrand noch sich selbst berühren darf.

Das Spiel bietet die Grundlage für eigene Erweiterungen und Verbesserungen. Sinnvoll erscheint zum Beispiel der Einbau mehrerer Levels oder auch die Möglichkeit zur Einstellung des Spieltempos. Den fertigen Film sowie das verwendete Material finden Sie wie immer auf der beiliegenden CD und unter [www.director7.de/workshop](http://www.director7.de/workshop).

Der Aufbau des Films erfolgt in mehreren Stufen, so daß Sie die einzelnen Abschnitte jeweils nacheinander anfertigen und testen können. Die Programmierung geht (im weiteren Sinne) objektorientiert vonstatten. An die Stelle eines komplexen Gesamtprogramms treten also einzelne, in sich mehr oder weniger geschlossene Programmmodule, die jeweils wenige spezielle Aufgaben erfüllen und über Nachrichten mit anderen Objekten kommunizieren.

Daß objektorientierte Programmierung in Lingo immer mit der Verwendung von Parentskripts einhergehen muß, ist ein weitverbreitetes Mißverständnis. Tatsächlich arbeitet Director selbst objektorientiert: So werden zum Beispiel von den Behaviors automatisch Objekte (oder Instanzen) im Speicher gebildet. Das Objekt ist quasi eine selbständige Kopie des Behaviors; es besitzt alle Prozeduren desselben und kann – völlig unabhängig von eventuell bestehenden weiteren Objekten des Behaviors – die objektbezogenen Daten in Variablen vom Typ Property speichern. Diese Variablen stehen dem Objekt für die Dauer seiner Existenz zur Verfügung. Der bedeutsamste Unterschied zwischen Behaviors und Parentskripts besteht darin, daß Sie sich um die Generierung und das Löschen der Objekte von Behaviors nicht selbst kümmern müssen: beides erfolgt automatisch. Die Lebensdauer eines Behavior-Objekts ist fest an die Existenz des entsprechenden Sprites im Drehbuch gebunden. Sobald der Abspielkopf bei der Wiedergabe das Sprite erreicht, werden von allen Behaviors des Sprites Objekte gebildet. Verläßt der Abspielkopf den Sprite oder stoppt der Anwender die Wiedergabe, löscht Director die Objekte wieder aus dem Speicher.

Ganz anders dagegen die Parentskripts. Ein Aufruf der Form

```
<Objektname> = new(script <Parentskriptname>)
```

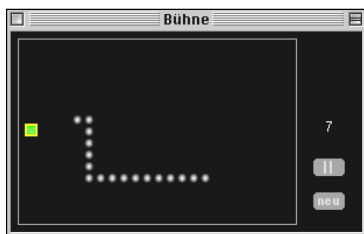
bildet die Objekte, so daß sie vollkommen unabhängig von der Position des Abspielkopfs existieren; auch nach dem Stopp der Filmwiedergabe, und zwar so lange, bis sie der Befehl

```
■ <Objektname> = 0
```

aus dem Speicher entfernt.

**Die Objektreferenz „me“.** Die Gemeinsamkeiten zwischen Objekten von Parentskripts und Behaviors zeigen sich auch bei einem Blick auf die Objektreferenz „me“. Dieses Keyword stellt innerhalb des Skripttextes eine Referenz zum Objekt selbst her. Theoretisch ließe sich „me“ auch durch einen beliebigen anderen Text ersetzen, es hat sich allerdings aufgrund der nicht zu übertreffenden Kürze und Eindeutigkeit des Begriffs eingebürgert. Wie Director Objektreferenzen speichert, zeigt Ihnen zum Beispiel das Frame-Skript in Abbildung 2. Sobald der Abspielkopf das Bild mit dem Skript „testbehavior“ erreicht, gibt Director den Wert von

Abbildung 1: Das Spiel „Snake“



Abbildungen 2: Die Objektreferenz „me“

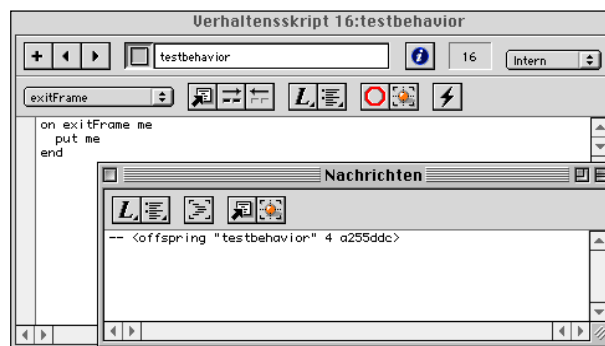
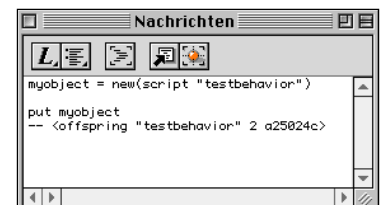


Abbildung 3: Objekt-Instanz bilden



„me“ im Nachrichtenfenster aus. Die Ausgabe enthält den Namen des zugrundeliegenden Skripts und die intern von Director verwendete Adresse. Verwenden Sie zum Vergleich der Einfachheit halber dasselbe Skript, um manuell eine Objekt-Instanz zu bilden (Abbildung 3). Beachten Sie dabei, daß Ihnen dieses Objekt im Unterschied zum Objekt des Behaviors auch nach dem Stopp der Filmwiedergabe zur Verfügung steht.

**Das Scripting im Überblick.** Das Spiel ließe sich prinzipiell in einem Bild des Drehbuchs realisieren. Unter dieser Voraussetzung ist es also nicht nötig, mit einem Parentskript zu arbeiten. Die Programmierung verteilt sich auf die folgenden Behaviors und Skripts:

- Das Film-Skript übernimmt lediglich die Aufgabe, die globalen Variablen zu initialisieren. Der Aufruf der dazu verwendeten Prozedur „initGlobals“ erfolgt automatisch vor dem Start der Wiedergabe des Films (Ereignis „prepareMovie“) und bei einem Klick auf den Button „Neu“.
- Das Behavior im Skriptkanal von Bild 2 sorgt für die Programmschleife im aktuellen Bild und für die Abfrage der Pfeiltasten.
- Im Behavior des Schlangenkopfs wird die Bewegung des Kopfs realisiert, wo-

bei dieses sowohl die Kollision mit der Spielfeldgrenze als auch das Erreichen der Mahlzeit prüft. Jedes gefundene Fressen erhöht den in der globalen Variablen „gCount“ enthaltenen Zähler, dessen Stand wiederum die Zahl der Segmente der Schlange und damit ihre Länge bestimmt. Hier erfolgt auch die Verwaltung der globalen Bühnenkoordinaten-Liste „gLocL“.

- Das Behavior der in den Kanälen 10 bis 522 angeordneten Segmente übernimmt die Positionierung des jeweiligen Segments anhand der globalen Koordinatenliste und prüft, ob es eine Berührung mit dem Kopf der Schlange gegeben hat, was zum Abbruch des aktuellen Spiels führt.
- Das Behavior der Mahlzeit ist für die zufällige Positionierung und die Animation der Graphik zuständig.
- Die Anzeige des aktuellen Punktestands übernimmt das Behavior des Textdarstellers, „count“.
- Mit den Behaviors der Buttons „pause“ und „neu“ schließlich wird das aktuelle Spiel unterbrochen respektive ein neues Spiel gestartet.

Die Behaviors kommunizieren untereinander durch das Versenden von Nachrichten, wozu der Befehl „sendSprite(<SpriteNummer>, [Parameter])“ dient. Um die Angabe fester Spritenummern in den Skripten zu vermeiden,

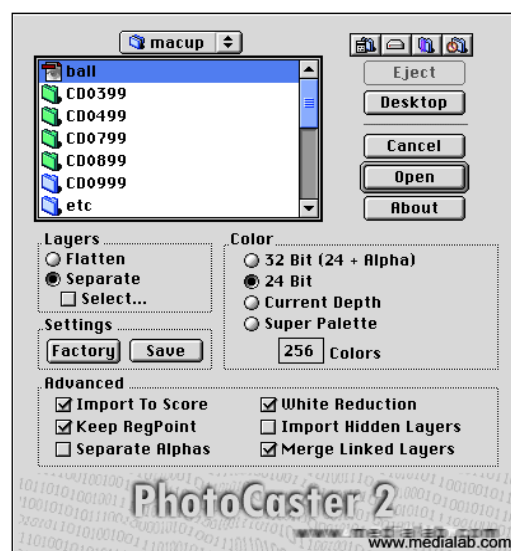
werden die in Frage kommenden Werte im Film-Skript als globale Variable definiert. Dies hat den Vorteil, daß man im Fall der Verschiebung eines Sprites im Drehbuch nur eine Stelle im Skript editieren muß.

**Der Film.** Erzeugen Sie zunächst einen neuen Film. Dann stellen Sie im Dialog „Modifizieren/Film/Eigenschaften“ eine Bühnengröße von 300 mal 170 Pixeln und im Feld „Drehbuchkanäle“ die Zahl 999 ein (Abbildung 4). Die Spielfeldumrandung erzeugen Sie mit dem Rechteckwerkzeug der Werkzeugpalette (Befehlstaste-7). Die exakte Größe von 240 mal 160 Pixeln geben Sie am besten mit Hilfe des Sprite-Inspektors oder der entsprechenden Felder im Drehbuch ein. Alternativ dazu kann die Einstellung auch im Dialog „Sprite-Eigenschaften“ (Befehls-Umschalttaste-I) erfolgen. Auch die beiden Buttons und der Textdarsteller zur Anzeige der Punktzahl werden mit der Werkzeugpalette erzeugt.

Für den Kopf und die Körpersegmente der Schlange verwenden wir ein und dieselbe Graphik. Dabei handelt es sich um einen in Photoshop erstellten Kreis, der mit einer weichen Auswahlkante versehen und mit einem Farbverlauf gefüllt wurde. Der Graphik-Import erfolgt mit dem Befehl „Ein- ➔

Abbildung 4: Film vorbereiten

Abbildungen 5: Graphikimport mit PhotoCaster



→ fügen/Media Lab Media“. Diesen Befehl stellt das Xtra „PhotoCaster“ zur Verfügung, das in einer Lite-Version zum Lieferumfang von Director 7 gehört (Abbildung 5). Die Vollversion ist unter [www.medialab.com](http://www.medialab.com) erhältlich.

Dieses Verfahren ermöglicht die Darstellung der Graphik mit weich auslaufender Kontur. Abbildung 6 zeigt die Ergebnisse des Imports mit PhotoCaster (links) und die des Standard-Imports ohne Verwendung des Alpha-Kanals (rechts) bei einer Vergrößerung von 400 Prozent. Auch ohne PhotoCaster ist der Import von Ebenen aus Photoshop-Dateien unter Beibehaltung der Transparenz möglich – wenn auch mit eingeschränktem Komfort. Schalten Sie dazu vor dem Import in Photoshop die Sichtbarkeit aller anderen Ebenen inklusive der Hintergrundebene aus. Die zur Animation der Mahlzeit benötigten Darsteller „egg.0“ bis „egg.3“ erzeugen wir mit dem Stift-Tool im Fenster „Malen“.

Den Aufbau des Drehbuchs und die Zuordnung der Kanäle zeigt Abbildung 7. Der Inhalt der nicht in der Abbildung enthaltenen Kanäle 21 bis 522 entspricht dem der Kanäle 10 bis 20. Hier ist der Darsteller „Layer 1“ mit dem „snake body“ genannten Behavior eingebunden. Erzeugen Sie diese Kanäle am besten durch Kopieren und Einfü-

gen. Da die Positionierung des Sprites später von dessen Behavior vorgenommen wird, verschieben Sie die Sprites anschließend in den nicht sichtbaren Bereich der Bühne. Dazu können Sie das Fenster „Schub“ (Befehlstaste-Umschalttaste-K) verwenden oder die entsprechenden Eingabefelder im Sprite-Inspektor respektive im Drehbuch.

**Bewegung der Schlange.** Die Schlange soll sich mit den Pfeiltasten nur in den vier Grundrichtungen bewegen lassen. Die Steuerung realisieren wir mit einem aus zwei Prozeduren bestehenden Frame-Skript. Während die Prozedur „exitFrame“ lediglich für das Verharren des Abspielkopfs im aktuellen Bild sorgt, wird die Ereignis-Prozedur „keyUp“ automatisch bei jedem Loslassen einer Taste aufgerufen und prüft den Code der gerade aktivierten Taste.

Eine Liste der Tastencodes ist mit dem Frame-Skript aus Abbildung 8 generierbar. Beachten Sie, daß die Bühne während der Wiedergabe des Films das aktive Fenster sein muß, damit Director die Tastatureingaben empfangen und interpretieren kann. Für die vier Pfeiltasten erhalten Sie Werte zwischen 124 und 126. Diese werden im Frame-Skript des Bilds 2 innerhalb der mit „case“ realisierten Programmverzweigung verwendet, um dem Sprite des Schlangen-

kopfs die Nachricht „move“ mit dem entsprechend gesetzten Richtungsparameter zu senden (Abbildung 9).

„keyUp“ läßt sich nicht dem Kopf-Behaviour zuordnen, da ein Versand von „keyUp“-Ereignissen an Sprites nur dann erfolgt, wenn diese mit einem editierbaren Textdarsteller verbunden sind.

Die Prozedur „move“ ist im Behavior des Kopfs (Abbildung 10) gespeichert und realisiert gemeinsam mit der Prozedur „enterFrame“ die eigentliche Bewegung. In Abhängigkeit vom übergebenen Richtungsparameter „direction“ setzt das Skript die Property „myDir“. Die If-Anweisungen verhindern dabei Wendungen um 180 Grad. Wenn sich die Schlange zum Beispiel gerade nach rechts bewegt, muß der Bewegung nach links ein Schwenk nach oben oder unten vorausgehen.

Den Test, ob der Schlangenkopf die Spielfeldgrenze erreicht hat, übernimmt die Prozedur „testBorder()“. Sie wird von der Prozedur „enterFrame“ aufgerufen und verwendet die Lingo-Funktion „inside“, um zu prüfen, ob sich die aktuellen Bühnenkoordinaten noch innerhalb des Rechtecks der Spielfeldbegrenzung befinden.

**Plazierung der Mahlzeit.** Die Mahlzeit soll eine zufällige Position erhalten und während des Spiels blinken. Die →

Abbildung 6

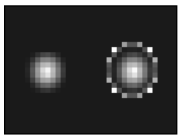
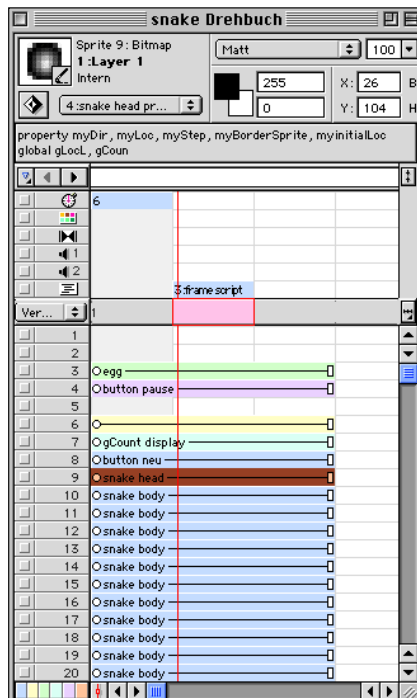
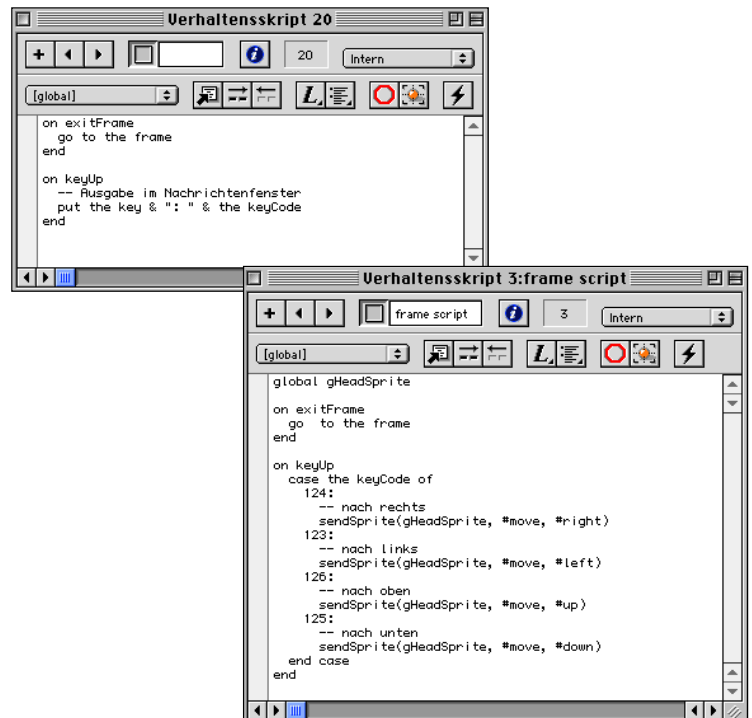


Abbildung 7: Drehbuchaufbau



Abbildungen 8 und 9: Tastencodes auflisten und im Skript festlegen



→ dafür erforderliche Animation erreichen Sie durch das Umschalten auf zwei weitere Graphikdarsteller. In der Prozedur „beginSprite“ werden zu diesem Zweck die Properties „myMem“, „myMemNum“ und „myMemCount“ gesetzt (Abbildung 11). Die Prozedur „enterFrame“ ruft im Spielmodus („gState“ hat den Wert „#play“) bei jeder Bewegung des Abspiekkopfs die Prozedur „animate“ auf, die für das zyklische Umschalten der Darsteller sorgt. Dazu wird der Inhalt der Property „myMemNum“ um 1 erhöht, sofern er das Maximum erreicht hat, auf den Wert 1 zurückgesetzt und anschließend zur Bildung des Darstellernamens in der Property „myMember“ verwendet. Die zufallsgenerierte Platzierung übernimmt die Prozedur „setLoc“. Sie ruft die im gleichen Behavior gespeicherte Prozedur „getaLoc“ auf, die mit Hilfe der Funktion „random()“ ein aus Zufallszahlen bestehendes Koordinatenpaar bildet. Als Start- und Maximalwert werden der Funktion „random()“ die Einzelwerte der Koordinaten der Spielfeldumrandung übergeben.

Da wir alle Positionen vermeiden wollen, die sich vollständig innerhalb der Schlange befinden, nehmen wir die Bildung der Zufallszahlen in einer Repeat-Schleife vor, die Director so lange ausführt, bis ein Koordinatenpaar gefunden ist, das außerhalb der Umrandung sämtlicher Segmente der Schlange liegt. Sobald es eine Berührung zwischen dem Kopf und der Mahlzeit gegeben hat, sendet das Behavior des Schlangenkopfs die Nachricht „disappear“. Die gleichnamige Prozedur schaltet dann lediglich auf den Darsteller „egg.0“ um.

**Das Wachstum der Schlange.** Die Länge der Schlange richtet sich nach der erzielten Punktzahl. Zu Spielbeginn besteht die Schlange aus zehn Segmenten. Dies erreichen wir durch das Setzen der globalen Variablen „gCount“ innerhalb der Prozedur „initGlobals“ im Filmskript. Mit jedem Punkt wächst die Schlange um ein weiteres Segment. Im vorliegenden Beispiel werden die Kanäle 10 bis 522 verwendet, um jeweils ein außerhalb der Bühne positioniertes

Segment des Körpers zu speichern. Die Schlange kann somit maximal aus 512 einzelnen Segmenten bestehen. Erreicht der Anwender diese Höchstgrenze, müßte in einer erweiterten Version dieses Spiels der nächste, schnellere Level starten.

Die neu angefügten Segmente machen natürlich alle Bewegungen der vorangegangenen Segmente mit. Damit alle Segmente dem Körper folgen können, ist es erforderlich, die vorausgegangenen Bühnenpositionen des Kopfs zu speichern. Diese Aufgabe übernimmt die globale Liste „gLocL“. Sie wird beim Start des Films durch die Prozedur „initGlobals“ im Filmskript mit 512 Elementen initialisiert und innerhalb der Prozedur „enterFrame“ des Kopf-Behaviors beschrieben:

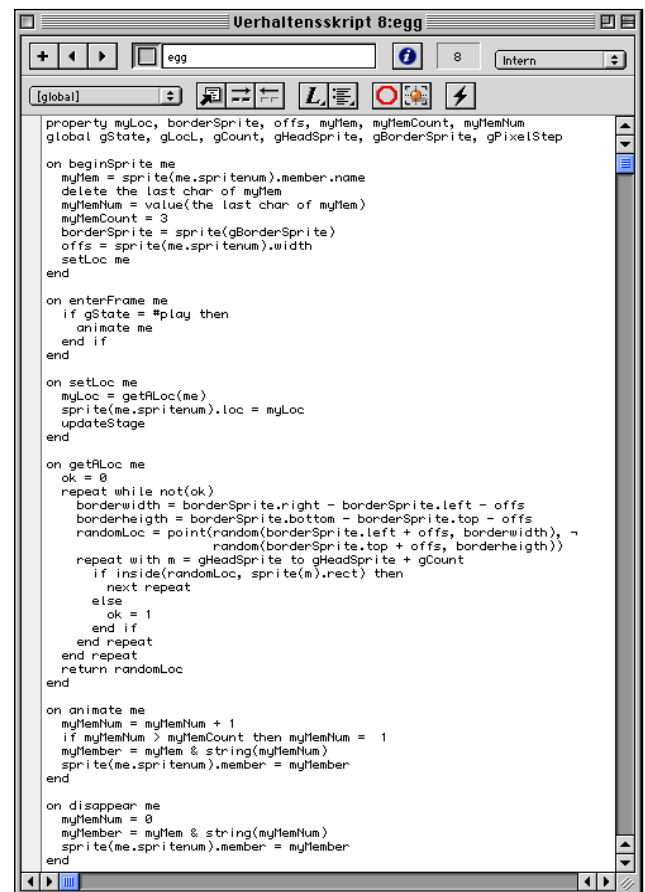
```
myLoc = myLoc + myDir
addat(gLocL, 1, myLoc)
deleat(gLocL, count(gLocL))
```

Bei jedem Wechsel der Bühnenposition setzt das Skript das aktuelle Koordinatenpaar an die Position 1 der

Abbildung 10: Bewegung generieren



Abbildung 11: Mahlzeit animieren



Liste und löscht gleichzeitig das letzte Element der Liste. Dadurch bleibt die Gesamtlänge der Liste stets konstant und jedes Koordinatenpaar durchläuft alle Positionen der Liste.

Da sich die Segmente in aufeinanderfolgenden Kanälen befinden, läßt sich die Kanalnummer als Grundlage zur Berechnung der Listenposition verwenden. Jedes Segment erhält seinen Listenindex, indem es von seiner aktuellen Kanalnummer diejenige des Kopf-Sprites subtrahiert. Das Ergebnis wird durch die Prozedur „beginSprite“ in der Property „myNumber“ gespeichert (Abbildung 12). Hier erfolgt auch das Ausblenden des Sprites mit Hilfe der Prozedur „hide“, welche den Sprite auf eine Bühnenposition außerhalb des sichtbaren Bereichs verschiebt.

Die Prozedur „enterFrame“ ist für die Anzeige des Segments zuständig. Sie prüft, ob sich der Film im Spielmodus „#play“ befindet und ob der aktuelle Zählerstand „gCount“ die Aktivierung des aktuellen Sprites erforderlich macht. Sind beide Bedingungen erfüllt, wird die neue Bühnenposition →

## BUCHTIP

■ Eine vollständige Einführung in Konzept und Design objektorientierter Anwendung hätte den Rahmen dieser Workshop-Folge gesprengt. Mehr zu diesem Thema finden Sie in dem Standardwerk für die objektorientierte Lingo-Programmierung:

→ *Peter Small: Lingo Sourcing – The Magic of Lists, Objects and Intelligent Agents; Wiley & Sons Ltd., ISBN 0-471-98615-1, Preis: 95,70 Mark*

■ Zusammen mit Joachim Gola hat unser Autor soeben das Buch „Director 7 Workshop“ herausgebracht, das anhand vieler Beispiele in die Programmierung mit dem Autorensystem einführt. Hier findet sich auch ein Kapitel über objektorientierte Techniken (das Inhaltsverzeichnis und ein Beispielkapitel gibt es als PDF unter [www.director7.de](http://www.director7.de)):

→ *Gerd Gillmaier, Joachim Gola: Director 7 Workshop – Professionelle Multimedia-Produktion für CD-ROM und Internet; Addison-Wesley, ISBN 3-8273-1443-7, Preis: 99,90 Mark*

→ des Sprites aus der globalen Liste gLocL gelesen und dem Sprite zugewiesen. Anschließend prüft die Prozedur bei allen Segmenten, die hinter dem zweiten Segment liegen, ob eines davon mit dem Sprite des Kopfs kollidiert ist. In diesem Fall endet das Spiel mit einer Meldung und dem Setzen der globalen Variable „gState“.

**Anzeige der Punktzahl.** Die Anzeige der aktuellen Punktzahl im Textdarsteller „count“ realisiert das dem Sprite des Textdarstellers zugeordnete Behavior (Abbildung 13). Die Prozedur „beginSprite“ definiert die Property „myText“ und weist ihr den aktuellen Zählerstand zu. Da der Film mit einem Anfangszählerstand „gStartCount“ von 10 startet, ist es nötig, diesen wieder

zu subtrahieren, bevor der Wert von „gCount“ angezeigt wird.

**Der Pause-Button.** Die Buttons erstellen wir mit dem Schaltflächentool der Werkzeugpalette. Das Behavior des Pause-Buttons (Abbildung 14) schaltet in der Prozedur „toggle“ den Spielmodus „gState“ um. Parallel dazu wird der im Button enthaltene Text modifiziert. Abhängig vom Spielmodus zeigt der Button entweder die Zeichen „>“ oder „||“ (zweimal Wahl taste-7). Ein Mausklick auf den Button veranlaßt das Behavior des Buttons „Neu“, die „toggle“-Prozedur aufzurufen.

**Der Button „Neu“.** Das Behavior dieses Buttons (Abbildung 15) stellt den Anfangszustand des Films wieder her. Es

ruft dazu die im Filmskript enthaltene Prozedur „initGlobals“ auf, die alle globalen Variablen initialisiert, und versendet die folgenden Nachrichten: An den Sprite des Kopfs wird „reset“ versandt, woraufhin dieser wieder die ursprüngliche Position einnimmt. Der Sprite der Mahlzeit erhält die Nachricht „setLoc“, erhält also eine neue, zufällige Position. Abschließend erfolgt an den Pause-Button der Aufruf „toggle“, wodurch sich das Spiel in den Modus „#pause“ setzt. *Gerd Gillmaier* ■

## VORSCHAU

■ In der nächsten Folge geht es unter anderem um die Steuerung von Animationen durch Lingo.

Abbildung 12: Schlangenwachstum

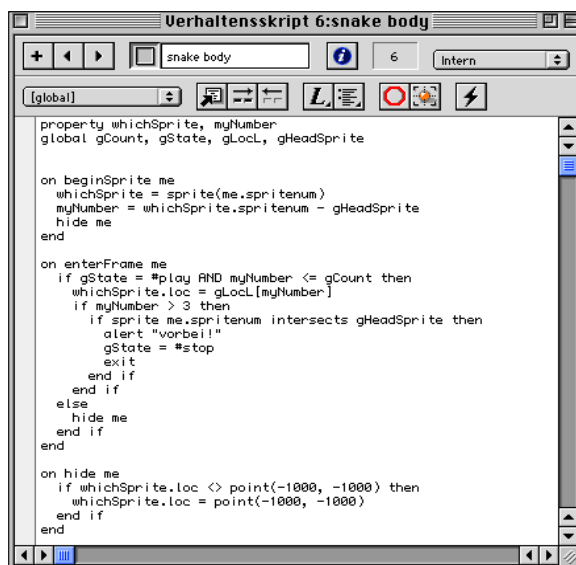
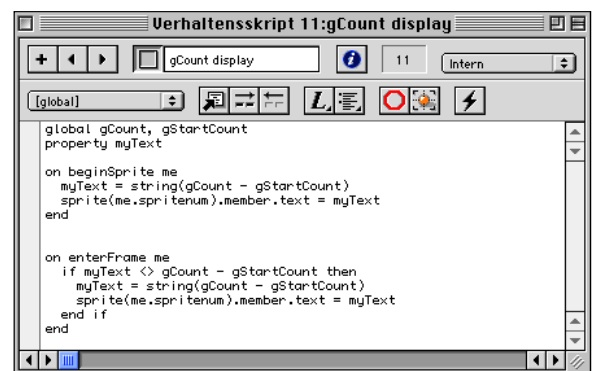


Abbildung 13: Punktezähler



Abbildungen 14: Pausen-Knopf

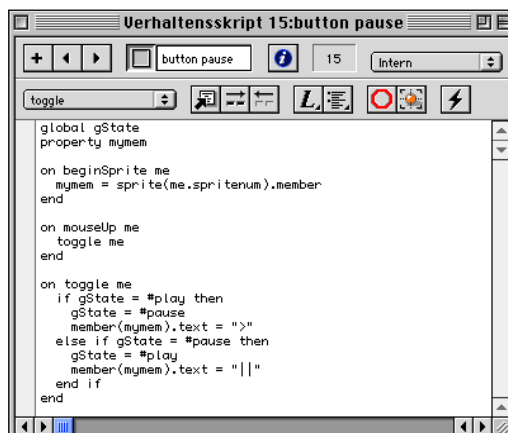


Abbildung 15: Knopf für neues Spiel

