sinclair

ZX Spectrum

T. Hartnell

Programme zum Lernen und Spielen

SYBEX

Gut verständliche Einführung in Maschinen code.

Sinclair ZX Spectrum

Sinclair ZX Spectrum

Programme zum Lernen und Spielen

T. Hartnell

SYBEX

Originalausgabe in Englisch.
Zuerst erschienen 1982 by Sinclair Browne Ltd., 10 Archway Close, London N 19 3TD
Titel der englischen Ausgabe: The ZX Spectrum Explored

Deutsche Übersetzung: Wolfgang Dederichs, Hattingen

Umschlaggestaltung und Satz: tgr – typo-grafik-repro gmbh., remscheid
Gesamtherstellung: Druckerei Hub. Hoch, Düsseldorf

ISBN 3-88745-022-1
1. Auflage 1983

Inhalt

Vorwort

Ein Blatt Papier erhält seine individuelle Bedeutung erst durch den Inhalt – ob es sich nun um ein Sonett von Shakespeare oder um die Kritzeleien eines Kindes handelt. Genauso verhält es sich mit dem Computer: So leistungsfähig der Computer auch sein mag, wirkliche Bedeutung erhält er erst durch das, was der Programmierer aus ihm macht. Daher wird klares und sorgfältiges Programmieren immer mehr an Bedeutung gewinnen. Die Aufgabe dieses Buches ist es, da anzuknüpfen, wo unser Spectrum-Handbuch aufhört. Kein Handbuch kann erschöpfend sein und unseres kann nur Grundlegendes über das Programmieren aussagen. Dieses Buch zeigt an Hand von Programmbeispielen, wie Sie fachgerecht programmieren; eine Fertigkeit, die Sie sich ohne diese Hilfe in jahrelanger, mühevoller Kleinarbeit aneignen müßten. Das vorliegende Buch eignet sich daher sowohl für den Leser, der gerade erst mit dem Programmieren angefangen hat, als auch für den erfahrenen Leser, der seine Programmierfähigkeiten noch erweitern möchte.

Danksagung

Eine Reihe von Bekannten hat mir geholfen, dieses Buch zu schreiben. Ich möchte ihnen herzlich dafür danken, daß sie mir ihre Erfahrung zur Verfügung stellten. Dadurch konnte ich dieses Buch weit verständlicher schreiben, als wenn ich mich ausschließlich auf meine eigene „Geschicklichkeit" verlassen hätte. Auch einige Bücher, die von Fall zu Fall am Ende eines Kapitels aufgeführt sind, haben sich als nützlich erwiesen und können dem Leser helfen, sein Wissen zu einzelnen Themen zu vertiefen. Beiträge zu spezifischen Kapiteln dieses Buches kamen von folgenden Mitarbeitern:

Mike Salem, Geschäftsführer von Hilderbay, einer der führenden Firmen, die für ZX kaufmännische Software herstellen, half mir bei der Einführung in das kaufmännische Kapitel und bei den fachlichen Erläuterungen im Anhang;

Jeff Warren, ein erfahrener Lehrer und Leiter der Software-Gesellschaft für Erziehung und Bildung CALPAC Computer-Software, half mir entscheidend beim Kapitel „Der Spectrum als Lehr- und Lernmittel". Fünf der Programme stammen aus seiner Feder.

Tim Rogers ist Student aus Richmond, der seiner Begeisterung für Computerspiele im Spiele-Kapitel freien Lauf ließ und der die Spiele Nächtlicher Überfall, Rebenklau, Meteoriten und Ausbruch schrieb.

Jerry Ruston, ein weiterer Student, wurde bisher mit drei Büchern bekannt: Pascal for Human Beings, The BBC Micro Revealed und The Book of Listings (eine BBC-Veröffentlichung, die er mit mir zusammen schrieb). Sein Beitrag zu diesem Buch ist das Kapitel über dreidimensionale graphische Darstellungen.

Dr. Tim Langdell, ein erfahrener Programmierer aus West Dulwich, der regelmäßig für die Zeitschriften „Your Computer" und „ZX Computing" schreibt und gegenwärtig ein Forschungsprojekt über intelligente Maschinen leitet, schrieb die Kapitel über Ton und Farbe.

James Walsh, ein Student aus Loughton, der gegenwärtig zwei Bücher schreibt (eines über Computerbau von Grund auf und ein anderes über den Gebrauch von Maschinencodes beim Spectrum), ist verantwortlich für die Beantwortung von Leserfragen in der Zeitschrift „Personal Computer World" und prüft die Software für meine eigene Zeitschrift „ZX Computing". Er schrieb im vorliegenden Buch das Kapitel über den Maschinencode.

Zum Schluß möchte ich noch Clive Sinclair danken, nicht nur dafür, daß er den Spectrum und seine Vorgänger erfand, ohne den es mir und hunderttausend anderen nicht gelungen wäre, die Computerwelt zu betreten, sondern auch für seine Ermutigung und tatkräftige Hilfe.

Einführung

Die Fertigstellung des Oxford University Press Lexikons dauerte 70 Jahre, weil seine Übersetzer absolut jedes englische Wort sowie alle Bedeutungen darin aufnehmen wollten. Mir wurde klar, daß es mir beim vorliegenden Buch über den *Sinclair ZX Spectrum* ähnlich ergehen könnte, wenn ich auf jede Möglichkeit, für die der Computer geschaffen ist, eingehen wollte. Deshalb habe ich mich darauf beschränkt, die wesentlichen Grundzüge des Programmierens beim Spectrum darzulegen und nur die Gebiete zu behandeln, auf denen Sie höchstwahrscheinlich Ihren Computer anwenden möchten.

Viele meiner Bücher enthalten die ständig wiederkehrende Bemerkung: „Dieses Buch ist zur praktischen Anwendung und nicht zum Lesen bestimmt." Damit will ich sagen, daß das Buch nicht wie ein Roman gelesen, sondern als Werkzeug zum unmittelbaren Gebrauch am Computer verwendet werden sollte. Sicher wird Ihnen das bloße Durchlesen schon etwas bringen, aber den richtigen Nutzen haben Sie erst, wenn Sie das Buch lesen, dabei den Spectrum eingeschaltet haben und jeweils die Programme eingeben, zu denen Sie im Text kommen.

Es besteht kein zwingender Grund, das Buch von A–Z in der vorliegenden Reihenfolge der Kapitel durchzuarbeiten. Es gibt sicher einiges, was Sie bereits wissen oder jetzt noch nicht wissen wollen. Überfliegen Sie dann in diesem Fall die betreffenden Kapitel, wenn Sie das erste Mal das Buch durcharbeiten, und kommen Sie später darauf zurück.

Wie auch immer, betrachten Sie dieses Buch über den *Sinclair ZX Spectrum* nicht als Lehrbuch. Lehrbücher sind oft abschreckend trocken, nehmen dem Leser jeden Spaß am Inhalt, und das wäre das letzte, was ich diesem Buch wünschen würde. Es ist eher als Wegweiser zum Computergebrauch zu verstehen, auf Gebieten, in denen wahre Abenteuer auf Sie warten. Mit einer solchen Einstellung können Sie viel Freude mit Ihrem Spectrum haben.

Zunächst werfen wir einen Blick auf die Grundlagen des Programmierens und gehen dann dazu über, Farbe und Ton ins Spiel zu bringen. Wir benutzen dabei Befehle, mit denen der Computer Effekte erzielt, an die wir nicht einmal im Traum gedacht hätten.

Als nächstes werden die Bereiche „Büroanwendungen" und „Erziehung" behandelt, jeweils mit einer Reihe ausführlicher Programme. Danach folgt ein größeres Kapitel über Spiele. Es enthält einige Spiele, die Sie sofort starten können, und soll Ihnen Ideen geben, wie Sie selbst Spiele entwickeln können. Im Spiele-Kapitel ist auch erklärt, wie der Benutzer eigene Grafiken erstellen kann.

Ein noch eindrucksvollerer Gebrauch der hochauflösenden Grafik des Spectrums ist im folgenden Kapitel über dreidimensionale Darstellungen erläutert.

Zum Schluß sehen wir uns den Maschinencode beim Spectrum an. Anschließend folgen noch einige Tips, wie Sie Ihren Programmierstil verbessern können.

Im Anhang befindet sich ein historischer Abriß über die Entwicklung von Computern, eine Erläuterung der damit verbundenen Fachausdrücke und eine Beschreibung der Besonderheiten des Spectrum.

Möge Ihnen dieses Buch in den kommenden Monaten ein verständlicher Führer und eine wesentliche Hilfe werden, um aus Ihrem Computer für Sie das Beste zu machen.

Viel Spaß beim Programmieren!

Tim Hartnell

London, im September 1982

Kapitel 1

Die Inbetriebnahme des ZX Spectrum

Wenn Sie Ihren Computer ausgepackt haben, werden Sie zunächst vielleicht etwas überrascht sein: Wie! Dieser kleine Kasten soll einen Computer enthalten und dazu noch mehr als 48000 Speicherplätze? Das sieht eher aus wie eine Schreibmaschinentastatur ohne Schreibwerk. Nachdem sich die erste Überraschung etwas gelegt hat, werden Sie sehen, was alles zum Gerät dazugehört:

Zunächst einmal brauchen Sie einen Fernseher, denn ohne einen Bildschirm können Sie nicht sehen, was der Computer alles macht und kann.

An der Rückseite des ZX Spectrum befinden sich eine Reihe von Steckbuchsen. Eine davon trägt die Aufschrift TV. Nehmen Sie jetzt das Kabel, welches auf der einen Seite in diese Steckbuchse paßt und auf der anderen Seite in den Antenneneingang Ihres Fernsehers gesteckt werden muß. Mit diesem Kabel wird die Verbindung zwischen Ihrem ZX Spectrum und dem Fernseher hergestellt. Der ZX Spectrum ist so ausgelegt, daß er über dieses Kabel Bilder an den Fernseher „sendet". Wenn Sie jetzt den Fernseher einschalten, sehen Sie nur Geflimmer auf dem Bildschirm. Ihr Spectrum sendet noch nichts; er ist noch nicht eingeschaltet.

Für den ZX Spectrum wird ein eigenes Netzgerät mitgeliefert. Auf der einen Seite hat es den normalen Netzstecker für 220 Volt, auf der anderen Seite geht das Verbindungskabel ab, welches den ZX Spectrum mit Strom versorgen soll. An der Rückseite rechts ist hierfür eine Steckbuchse vorgesehen, die mit „9V DC" beschriftet ist. (Achtung: Wenn Sie noch einen Drucker oder ein weiteres Gerät gekauft haben, stellen Sie diese Geräte bitte erst einmal auf die Seite. Auf keinen Fall dürfen Sie diese Geräte an den ZX Spectrum anschließen, wenn er unter Strom steht. Also immer erst den Netzstecker herausziehen!)

Sobald der ZX Spectrum über das Netzgerät mit dem Stromnetz verbunden ist, sendet er ein Bild an den Fernseher auf Kanal 36. Sie können jetzt

versuchen, Ihren Fernseher auf den richtigen Empfang einzustellen. Es muß dann unten der Text „© 1982 Sinclair Research Ltd." erscheinen. Begnügen Sie sich zunächst einmal mit diesem Bild, selbst wenn es Ihnen etwas unscharf vorkommt. Wenn Sie längere Zeit mit dem Computer experimentiert haben, werden Sie genügend Gelegenheit haben, die für Sie richtige Einstellung bei vollem Bildschirm zu finden.

Tippen Sie jetzt ruhig mal ein wenig auf der Tastatur herum. Auf dem Bildschirm erscheinen dann eine Reihe von Zeichen und Wörtern. Anfangs werden die gummiartigen Tasten für Sie noch etwas ungewohnt sein. Aber nach einiger Übung dürfte dies für Sie schon leichter von der Hand gehen. Vielleicht haben Sie schon bemerkt, daß bei einigen Tasten gleich ganze Wörter erscheinen. Diese Eigenschaft wird für das Programmieren später eine sehr wesentliche Erleichterung darstellen. Wenn Sie bei Ihren Tipp-Übungen zuviel Durcheinander auf dem Fernsehbildschirm erzeugt haben, können Sie den Rechner ruhig einmal aus- und wieder einschalten und noch ein paar Experimente durchführen.

Der ZX Spectrum als Taschenrechner

Einiges wird Ihnen sicherlich bekannt vorkommen. Tippen Sie einfach mal die folgenden Tasten (das + erhalten Sie, wenn Sie die SYMBOL SHIFT-Taste gedrückt halten und dazu die Taste tippen, auf der das +-Zeichen steht):

```
PRINT
7
+
6
ENTER
```

und schon liefert Ihnen der Computer das Ergebnis: nämlich 13. Er hat also wie ein Taschenrechner funktioniert. Sie mußten lediglich vorweg das Kommando PRINT eingeben und hinterher statt des gewohnten Gleichheitszeichens ein ENTER. Versuchen Sie mal einige andere Beispiele und geben Sie statt 7 + 6 die folgenden Aufgaben ein:

```
PRINT 1 + 4 ENTER
PRINT 8 / 2 ENTER
PRINT 12 / 4 ENTER
PRINT 3 * 4 * 5 ENTER
```

Der ZX Spectrum läßt sich also wie ein normaler Taschenrechner verwenden.

Wir können die Gelegenheit benutzen und rasch ein paar Aufgaben lösen, die uns im täglichen Leben immer wieder begegnen: Z. B. soll die

Mehrwertsteuer zu einem Nettobetrag von 250 DM berechnet werden. Bei einem Mehrwertsteuersatz von 14% ergibt das

```
PRINT 250 * 14 ENTER
```

Dieser Betrag muß noch durch 100 geteilt werden. Die Mehrwertsteuer beträgt dann 35 DM.

Als nächstes sollen die Zinsen berechnet werden, die Sie für 250 DM bei einem Zinssatz von 5% nach einem Jahr erhalten:

```
PRINT 250 * 5 ENTER
PRINT 1250/100 ENTER
```

Sie erhalten also nach einem Jahr 12,50 DM Zinsen.

Als letztes Beispiel soll der Rechner noch die Summe der Zahlen von 1–10 bilden:

```
PRINT 1+2+3+4+5+6+7+8+9+10 ENTER
```

Das Ergebnis lautet 55.

Wenn Sie sich vertippt haben, hilft im Moment nur, den Rechner auszuschalten und alles noch einmal zu tippen. Wie wir uns weiter helfen können, werden wir im weiteren Verlauf noch sehen.

Der Anschluß eines Kassettenrecorders

Auf der Rückseite des ZX Spectrum befinden sich zwei Steckbuchsen mit der Aufschrift „EAR" und „MIC". Die Buchse „EAR" muß mit dem entsprechenden Ausgang am Kassettenrecorder verbunden werden. Die Verbindung der Buchse „MIC" mit dem Mikrofoneingang am Recorder benötigen wir noch nicht.

Wir sind jetzt in der Lage, Programme von Kassette zum ZX Spectrum zu übertragen. Bei einigen Geräten wird eine Beispielkassette mitgeliefert. Diese enthält zwei Programme:

ein Programm mit Schreibübungen sowie
ein Programm mit einigen Spielen.

Um ein Programm in den Rechner zu holen, müssen wir dem Rechner „Bescheid sagen". Mit den Angaben

```
LOAD "" ENTER
```

sagen wir dem Rechner, daß gleich vom Kassettenrecorder ein Programm überspielt wird. Schalten Sie jetzt bitte den Recorder ein und spielen Sie

die Kassette ab. Der ZX Spectrum analysiert die Signale, die vom Kassettenrecorder kommen, und baut die Signale in Programmbausteine um. Auf dem Bildschirm erscheinen eine Reihe von Streifen, die erst wieder verschwinden, wenn das Programm vollständig im Rechner angekommen ist. Vielleicht haben Sie beim ersten Mal Pech mit der Übertragung. Das Laden von Programmen klappt nicht immer auf Anhieb. Die Klangfarbe sollte auf hell eingestellt sein und die Lautstärke etwa auf 2/3 stehen. Eventuell sind mehrere Versuche erforderlich, bis Sie die richtige Einstellung herausgefunden haben. Wenn das Programm einwandfrei geladen wurde, befindet es sich jetzt im Rechner und kann mit

```
RUN ENTER
```

gestartet werden.

Nun wollen Sie sicher noch wissen, wie das Programm auf die Kassette gekommen ist. Verbinden Sie hierzu bitte die Buchse mit der Aufschrift „MIC" mit dem Mikrofoneingang Ihres Kassettenrecorders. Bei einigen Recordern empfiehlt es sich, den anderen Anschluß (EAR) solange auszustöpseln. Tippen Sie jetzt

```
SAVE "NAME"
```

Bevor Sie ENTER tippen, schalten Sie bitte Ihren Recorder auf Aufnahme. Danach überspielt der ZX Spectrum das Programm in Form von Tonsignalen zum Kassettenrecorder. Wieder erscheinen Balken auf dem Fernseher, bis die Übertragung abgeschlossen ist. Wenn Sie jetzt hören wollen, was der Computer übertragen hat, können Sie sich das Stück mal vorspielen.

Damit sind wir in der Lage, die Programme aus diesem Buch auch auf Kassetten zu speichern und nach Belieben wieder in den Rechner zu laden. Sie können selbstverständlich auch mehrere Programme hintereinander auf die Kassette packen. Wenn Sie jedem Programm einen Namen geben, können Sie später gezielt mit dem Befehl

```
LOAD "NAME" ENTER
```

das gewünschte Programm wieder von der Kassette in den Rechner holen.

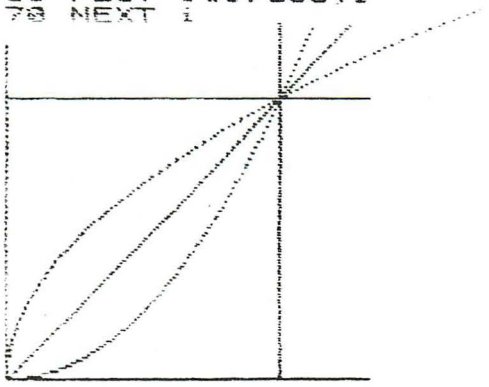
Der Anschluß des ZX Druckers

Bevor Sie jetzt den Drucker anschließen, müssen Sie bitte erst den ZX Spectrum ausschalten! Andernfalls könnte er durch einen Kurzschluß beschädigt werden. Also erst den Netzstecker ziehen! Jetzt können Sie Ihren Drucker hinten an den Spectrum anschließen.

Zur Grundausstattung des Druckers gehört eine Rolle Papier. Hierbei handelt es sich um aluminiumbeschichtetes Spezialpapier. Beim Druckvorgang werden aus der Aluminiumschicht Punkte durch elektrischen Kontakt herausgebrannt.

Wir können unsere Rechenexperimente von vorhin wiederholen. Alles, was oben auf dem Bildschirm erscheint, kann auf dem Drucker festgehalten werden. Wir brauchen dazu nur die Taste COPY (unten links) und anschließend die Taste ENTER zu tippen. Damit ist der Drucker einsatzbereit und wir können jederzeit alles auf dem Papier festhalten, was augenblicklich auf dem Bildschirm erschienen ist. Das gilt auch für Bildschirmgrafik!

```
10 FOR i=1 TO 176
20 PLOT i,i
30 PLOT 0,i: PLOT 100,i
40 PLOT i,0: PLOT i,100
50 PLOT i,i*i/100
60 PLOT i*i/100,i
70 NEXT i
```



Kapitel 2

Programmieren in BASIC

BASIC ist die weltweit bekannteste Programmiersprache, weil diese Sprache sehr leicht zu lernen ist. Computersprachen werden in verschiedene Kategorien eingeteilt: „Höhere" Sprachen stehen der menschlichen Ausdrucksweise näher, während Programmiersprachen auf der unteren Ebene mehr maschinenbezogen arbeiten und der Arbeitsweise des Computers nahekommen, der aus einer Fülle verdrahteter Nullen und Einsen besteht.

BASIC ist eine „höhere" Sprache. Selbst wenn Sie zu diesem Zeitpunkt keine Programmier-Erfahrung haben, werden Sie sicher erleichtert sein, wenn Sie hören, daß Sie bereits ein bißchen BASIC können:

Wörter wie PRINT (Drucken), STOP und LIST (Liste) bedeuten das gleiche in BASIC wie im Englischen. Ebenso ist es mit den Wörtern IF (falls), THEN (dann) und OR (oder). Wie Sie sehen, ist Programmieren einfacher, als Sie gedacht haben.

Wenn Sie ein Programm schreiben, geben Sie dem Spectrum eine Anzahl Befehle, die er ausführen soll. Wie alle Computer heute, hat der Spectrum erstaunliche Fähigkeiten, zu rechnen und Entscheidungen zu treffen, aber absolut kein eigenes Denkvermögen. Wenn Sie dem Computer etwas befehlen, führt er es aus. Wenn Sie einen Teil der Befehle auslassen, wird er versuchen den Rest zu befolgen, ohne zu merken, daß etwas fehlt.

Stellen Sie sich vor, Sie hätten einen Roboter als Diener und er sollte Ihnen Badewasser einlassen. „Geh ins Badezimmer" könnte der erste Befehl in seinem Programm lauten. „Wenn die Badewanne leer ist, dann drehe den Wasserhahn auf. Wenn die Badewanne voll ist, drehe den Wasserhahn zu." Der Roboter würde glücklich ins Badezimmer schreiten, seine elektronische Hand in die Badewanne stecken, feststellen, daß sie leer ist und den Wasserhahn aufdrehen. Er würde dann stehenbleiben und warten, bis das Wasser eingelaufen ist, und dann den Wasserhahn wieder

zudrehen. Aber weil Sie den Befehl „Schau nach, ob der Stöpsel im Ausfluß ist" und „Falls der Stöpsel nicht im Ausfluß ist, stecke den Stöpsel in den Ausfluß" vergessen haben, denkt der Roboter nicht von sich aus daran.

Genauso arbeitet der Spectrum. Er befolgt nur eingegebene Befehle. Die Befehle, die der Spectrum versteht, sehen so ähnlich aus, wie die Befehle, die der Roboter ausführen sollte, allerdings auf Englisch.

Hier ist ein einfaches Programm für den Spectrum, das Sie sicherlich auch ohne Programmier-Erfahrung verstehen werden:

```
LET A = 20
LET B = A + A
IF B = 40 THEN PRINT "B ist gleich 40"
```

Das klingt zwar wie einfaches Englisch, aber es ist auch BASIC. Die erste Zeile des Programms liest sich etwa so: Setze A = 20; die zweite heißt dann: Setze B auf das doppelte. Die dritte Zeile ähnelt einem Befehl an den Roboter: Falls (IF) Badewanne leer, dann (THEN) drehe Wasserhahn auf; falls (IF) B = 40 dann (THEN) gib aus „B ist gleich 40". Sie könnten dieses Programm fast schon Ihrem Spectrum eingeben.

Sie müssen nur noch eine Zahl vor jede Zeile setzen, damit aus ihnen ein Programm wird. Das sieht dann so aus:

```
10  LET A = 20
20  LET B = A + A
30  IF B = 40 THEN PRINT "B ist gleich 40"
```

Den einzelnen Zeilen kann jede beliebige Zahl (zwischen 1 und 9999) zugeordnet werden. Sie werden vom Computer automatisch in der richtigen Reihenfolge sortiert. Ein guter Rat lautet, die Zeilennummern in Zehnerschritten zu vergeben. So bleibt noch genügend Platz, um später einzelne Zeilen zwischenzufügen.

Jetzt haben Sie schon etwas Übung mit den BASIC-Ausdrücken LET, IF, THEN und PRINT. Das Gleichheits- und das Pluszeichen kennen Sie sicher aus der Mathematik. Sie haben beim Programmieren mit BASIC meistens die gleiche Bedeutung wie beim Rechnen.

Bevor wir etwas programmieren, sollten Sie noch einen weiteren BASIC-Ausdruck kennenlernen. Er besteht eigentlich aus zwei Wörtern und wird Ihnen schon bekannt sein: GO und TO.

In BASIC werden sie immer zusammen verwendet als GO TO und mit einer Taste (der G-Taste) dem Spectrum eingegeben. Sie erinnern sich, wir leiteten unser „Badewasser-einlassen-Programm" mit dem Befehl „Gehe

ins (GO TO) Badezimmer" ein. Beim Programmieren geben wir den Befehl: Gehen Sie zu einer Zeile mit der Nr. . ., also z. B. GO TO 10.

Wir können unserem dreizeiligen Programm (s. o.) noch eine Zeile 40 hinzufügen:

```
40  GO TO 30
```

Jetzt würde der Computer erst Zeile 30 ausführen und „B ist gleich 40" anzeigen. Dann würde er zur Zeile 40 weitergehen und dort den Befehl GO TO 30 vorfinden. Ohne nach Gründen zu fragen, würde er „GO TO 30" ausführen, nach Zeile 30 gehen und erneut „B ist gleich 40" anzeigen. Dann würde er wieder nach Zeile 40 vorrücken, wo er wieder den Befehl „GO TO 30" ausführen würde usw. bis in alle Ewigkeit, oder bis der Bildschirm voll wäre mit den Worten „B ist gleich 40".

Jetzt wird es Zeit, daß wir den Spectrum wieder anschalten.

Die Tastatur des Spectrum sieht sonderbar aus! All diese merkwürdigen Wörter wie MERGE und VERIFY und all die mathematischen Ausdrükke, wie SIN und COS! Aber keine Angst: Was Sie in der Schule gelernt (und vielleicht inzwischen vergessen) haben, kommt jetzt wieder. Sie werden sich sehr schnell mit der Tastatur auskennen und, wie es in der Werbung für den Vorgänger des Spectrum heißt, „sich bald mit ihm wie mit einem alten Freund unterhalten".

Die Tastatur

Zwei der wichtigsten Bestandteile der Tastatur sind die Shift-Tasten. Sie befinden sich auf der unteren Reihe: die CAPS SHIFT-Taste (weiß) finden Sie in der linken unteren Ecke, und die SYMBOL SHIFT-Taste (rot) ist die zweite von rechts in der rechten unteren Ecke. Von der Schreibmaschine her werden Sie die Funktionsweise dieser Tasten schon kennen. Diese Tasten entscheiden darüber, was aus der verwirrenden Fülle von Wörtern und Symbolen auf dem Bildschirm angezeigt werden soll.

Tippen Sie jetzt, nachdem der ZX Spectrum eingeschaltet ist, irgendeine weiße Buchstabentaste. Auf dem Bildschirm erscheint das Wort, welches auf der Taste steht (z. B. LOAD, LIST, PRINT). Solch ein Wort wird Schlüsselwort genannt. Das System der Schlüsselwörter ist eine der Ursachen, weshalb der Spectrum einen so guten Gebrauch von seinem Speicher macht. Diese Eigenschaft macht das Programmieren so leicht. Indem Sie eine Taste drücken, haben Sie schon das ganze Wort eingegeben. Bei den meisten anderen Computern müssen Sie Worte wie LIST, DIM oder FOR voll ausschreiben.

Drücken Sie jetzt gleichzeitig die CAPS SHIFT- und die DELETE-Taste (in der oberen rechten Ecke), bis alles, was Sie bis jetzt geschrieben haben, gelöscht ist.

Drücken Sie jetzt die P-Taste. Es erscheint das Wort PRINT. Drücken Sie dann einige beliebige Zahlen-Tasten, und Sie sehen unten auf dem Bildschirm den Ausdruck PRINT in Verbindung mit einer Zahl, z. B. PRINT 62735. Drücken Sie jetzt die Taste mit der Aufschrift ENTER. Der Bildschirm wird gelöscht und die gesuchte Zahl erscheint oben auf dem Bildschirm. Jetzt erst hat der Spectrum den Befehl ausgeführt und eine Zahl ausgegeben (PRINT). Bevor der Spectrum irgendetwas tut, wartet er nämlich so lange, bis Sie die ENTER-Taste gedrückt haben.

Wir geben jetzt ein einfaches Programm ein, um zu demonstrieren, wie das System der Schlüsselwörter benutzt wird. Vorab tippen Sie bitte die CAPS SHIFT-Taste und dazu die Taste 2. Hierdurch wird die Tastatur generell auf Großbuchstaben (CAPital letterS) eingestellt:

```
10 INPUT (drücken Sie die I-Taste) A (drücken Sie jetzt ENTER,
   um anzuzeigen, daß Sie die Programmzeile abgeschlossen haben)
20 PRINT A (drücken Sie ENTER)
30 GO TO 10 (drücken Sie ENTER)
```

Merken Sie sich, daß das Wort PRINT mit der P- und der Begriff GO TO mit der G-Taste geschrieben wird. Die Leerstellen werden vom ZX Spectrum automatisch eingebaut. Wenn Sie das alles in Ihren Computer eingegeben haben, geben Sie ihm mittels der R-Taste das Signal zum Start – das Wort RUN erscheint auf dem Bildschirm – und drücken Sie dann ENTER. Ein blinkender Cursor erscheint unten auf dem Bildschirm als Zeichen dafür, daß der Computer auf eine Zahl wartet. Geben Sie eine Zahl ein und drücken Sie dann ENTER. Jetzt sehen Sie Ihre Zahl oben auf dem Bildschirm. In Erinnerung an die „GO TO Badezimmer"-Diskussion vorhin wissen Sie, daß die Schlußzeile des Programms (Zeile 30) den Computer zur Zeile 10 zurückgehen läßt. Wenn Sie ihn nicht stoppen, macht er jetzt immer weiter. Sie können ihn aber auch durch einen Trick stoppen: Geben Sie statt einer Zahl irgendeinen Buchstaben (außer A) ein. Dann schreibt der Computer unten auf dem Bildschirm eine Fehlermeldung aus und stoppt.

Jetzt tippen Sie die A-Taste, drücken ENTER und der Bildschirm wird wieder frei. Nur die Copyright-Notiz unten auf dem Bildschirm bleibt stehen. Das Schlüsselwort der A-Taste heißt NEW (neu) und löscht im Computer alles bis dahin Gespeicherte. Deshalb müssen Sie diese Taste mit größter Vorsicht benutzen.

Jetzt wissen Sie, daß Sie die weißen Schlüsselwörter durch einen entsprechenden Tastendruck schreiben können, wenn Sie vorher eine Zeilennummer eingegeben haben.

Nun zu den roten Begriffen auf den Tasten. Drücken Sie gleichzeitig die SYMBOL SHIFT- und die Y-Taste. Das Wort AND erscheint auf dem Bildschirm. Drücken Sie die G-Taste gleichzeitig mit der SYMBOL

SHIFT-Taste und es erscheint das Wort THEN. Soviel zu den roten Angaben auf den Tasten. Zeichen wie +, ' und $ bekommen Sie, wenn Sie gleichzeitig SYMBOL SHIFT und die Taste des gewünschten Zeichens drücken.

Die grünen Wörter über den Tasten werden eingeschaltet, wenn Sie beide SHIFT-Tasten gleichzeitig drücken, wieder loslassen und dann erst die gewünschte Wort-Taste berühren. Versuchen Sie es! Wenn Sie zuerst CAPS SHIFT- und SYMBOL SHIFT-Taste zugleich drücken, wieder loslassen und danach die A-Taste drücken, müßte das Wort READ erscheinen.
Etwas schwieriger ist es, die roten Wörter unter den Tasten zu erhalten. Drücken Sie gleichzeitig beide SHIFT-Tasten, lassen Sie dann CAPS SHIFT los, aber halten Sie SYMBOL SHIFT weiter heruntergedrückt. Drücken Sie dann die X-Taste. Das Wort INK erscheint. Üben Sie das ein wenig, um sicherzugehen, daß das Wort INK (oder BEEP, PAPER, FLASH oder BRIGHT) auch immer erscheint. Das gleichzeitige Drükken der beiden Shift-Tasten dient also als einmaliger Umschalter für die gesamte Tastatur. Es gelten dann nur die über bzw. unter der jeweiligen Taste angegebenen Bezeichnungen.

Lesen Sie sich noch einmal den ganzen Abschnitt durch und wiederholen Sie jede Übung, bis Sie alles verstanden haben. Zwar wird Ihnen alles zunächst sehr kompliziert und langwierig vorkommen, aber Sie werden dann doch überrascht sein, wie schnell Sie das Tastatursystem begriffen haben. Dieses Einführungskapitel schließt mit einem Programm, das Ihnen beibringt, wo sich die Buchstaben-Tasten auf dem Spectrum befinden. Es soll Sie zum Tippen ermutigen. Doch vorher sollen Sie das erste wirkliche Programm in diesem Buch vorgestellt bekommen. Auch wenn das Eintippen erst einmal viel Zeit beansprucht, halten Sie bitte durch. Nach der Eingabe des ganzen Programms drücken Sie RUN und spielen ein Spiel mit dem Spectrum.

Das Programm STEIN, SCHERE, PAPIER ist eine Computerversion des Spiels, in dem zwei Spieler erst eine Hand hinter ihrem Rücken verstecken und wenn sie sie wieder zeigen, ein Symbol für Stein (eine geschlossene Faust), für Schere (zwei ausgestreckte Finger) oder für Papier (flache offene Hand) zeigen. Der Stein schlägt die Schere (weil er sie schleifen kann), die Schere schlägt das Papier (weil sie es schneiden kann) und das Papier schlägt den Stein (weil es ihn einwickeln kann). Mit den Zahlen 1, 2, oder 3 geben Sie ein, ob Sie Stein, Schere oder Papier wählen. Das gewünschte Wort erscheint dann oben auf dem Bildschirm.

Noch ein paar nützliche Ratschläge, bevor Sie das Programm eingeben: Sie erhalten das kleine Copyright-Zeichen in Zeile 20, indem Sie erst beide SHIFT-Tasten drücken, dann die SYMBOL SHIFT-Taste unten halten und gleichzeitig die P-Taste drücken. Das Gleichheitszeichen (ab Zeile 30) erhalten Sie, wenn Sie die SYMBOL SHIFT-Taste herunterdrük-

ken, festhalten und dann die L-Taste drücken. Denken Sie daran, nach jeder Programmzeile ENTER zu drücken. Nur so kann die Zeile vom unteren Teil des Bildschirms nach oben ins Programm rutschen. Wenn das nicht klappt, erscheint ein flackerndes Fragezeichen irgendwo in der Zeile. Damit signalisiert der Computer, daß Sie in der Zeile einen Fehler gemacht haben. Suchen Sie an der Stelle, an der das Fragezeichen aufflakkert, nach dem Fehler. Sie können mit den Pfeil-Tasten in der Zeile nach links oder rechts zu der fehlerhaften Stelle gehen, den Fehler mit DELETE beseitigen, Ihre Berichtigung eingeben und danach erneut ENTER tippen.

Der Doppelpunkt (:) nach dem Wort CLS (Clear Screen = Löschen Bildschirm) in Zeile 60 befindet sich auf der Z-Taste. Drücken Sie dafür gleichzeitig die SYMBOL SHIFT- und die Z-Taste. Zeile 70 enthält ein einzelnes Apostroph (') nach dem Wort PRINT, vor dem Anführungszeichen ("). Für das einzelne Apostroph müssen Sie gleichzeitig SYMBOL SHIFT und die Taste 7 drücken. Die Anführungszeichen erhalten Sie ebenso mittels der P-Taste.

Vorsicht, die Zeile 170 ist etwas schwierig in den Computer zu bekommen. Nach 170 LET C = brauchen Sie das Wort INT. Tippen Sie nicht Buchstabe für Buchstabe ein, sondern holen Sie es von der R-Taste. Drücken Sie beide SHIFT-Tasten, lassen Sie sie los, drücken Sie dann R, und INT erscheint auf dem Bildschirm. Die offene Klammer befindet sich auf Taste 8 (drücken Sie gleichzeitig SYMBOL SHIFT und Taste 8), und das Wort RND erhalten Sie von der T-Taste. Drücken Sie beide Tasten gleichzeitig, lassen Sie sie los, und drücken Sie dann T. Das Sternchen (*), mit dem in BASIC multipliziert wird, kommt von der B-Taste. Sie erhalten es, indem Sie SYMBOL SHIFT gleichzeitig mit der B-Taste drücken.

Die Wörter AND, OR, THEN und TO erhalten Sie, indem Sie die SYMBOL SHIFT-Taste gleichzeitig mit der in Frage kommenden Buchstabentaste drücken. Geben Sie jetzt das Programm ein, bevor Sie weiter den Text durchgehen.

```
  10 REM STEIN, SCHERE, PAPIER
  20 REM © HARTNELL, 1982
  30 LET comp=0
  40 LET spi=0
  50 FOR a=1 TO 10
  60 CLS: PRINT "RUNDE NUMMER ";
a
  70 PRINT '"1 - STEIN"
  80 PRINT "2 - SCHERE"
  90 PRINT "3 - PAPIER"
 100 PRINT '"BITTE WAEHLEN: 1,2
ODER 3!"
 110 INPUT b
 120 PRINT '"SIE WAEHLEN ";
 130 IF b=1 THEN PRINT "STEIN"
```

```
 140 IF b=2 THEN PRINT "SCHERE"
 150 IF b=3 THEN PRINT "PAPIER"
 160 PAUSE 50
 170 LET c=INT (RND*3)+1
 180 PRINT '"ICH WAEHLE ";
 190 IF c=1 THEN PRINT "STEIN"
 200 IF c=2 THEN PRINT "SCHERE"
 210 IF c=3 THEN PRINT "PAPIER"
 220 LET d=260
 230 IF b=c THEN PRINT '"UNENTSC
HIEDEN !!!": GO TO d
 240 IF c=1 AND b=2 OR c=2 AND b
=3 OR c=3 AND b=1 THEN PRINT '"I
CH GEWINNE !!!": LET comp=comp+1
:GO TO d
 250 IF b=1 AND c=2 OR b=2 AND c
=3 OR b=3 AND c=1 THEN PRINT '"S
IE GEWINNEN!!!": LET spi=spi+1
 260 PRINT '"PUNKTE"''"ICH> ";co
mp;TAB 10,"SIE> ";spi
 270 PAUSE 100
 280 NEXT a
 290 IF spi=comp THEN PRINT ''"D
AS SPIEL WAR UNENTSCHIEDEN!"
 300 IF spi<comp THEN PRINT ''"I
CH HABE DIESES SPIEL GEWONNEN!"
 310 IF comp<spi THEN PRINT ''"S
IE HABEN DIESES SPIEL GEWONNEN!"
```

So sieht dann beispielsweise der Spielverlauf aus:

```
RUNDE NUMMER 7

1 - STEIN
2 - SCHERE
3 - PAPIER

BITTE WAEHLEN: 1,2 ODER 3

SIE WAEHLEN PAPIER

ICH WAEHLE SCHERE

ICH GEWINNE!!!

PUNKTE

ICH> 2      SIE> 3
```

Zu jeder Programmzeile folgt jetzt eine kurze Erläuterung:

10 Diese Zeile startet mit REM, was soviel wie Bemerkung (REMark) bedeutet. REM signalisiert, daß alles, was sich direkt daran anschließt, nur den Leser angeht; der Computer übergeht alles, was auf das Wort REM folgt.

20 Auch dies ist eine REM-Bemerkung. Sie wird ebenfalls nicht beachtet.

30 LET bedeutet in BASIC etwa soviel wie „setze". Das Wort COMP (Kurzform von Computer) wird als Variable verwendet. Variablennamen setzen sich aus einer beliebigen Kombination von Buchstaben und Zahlen zusammen (angefangen mit einem Buchstaben). Der Variablen COMP wird ein numerischer Zahlen-Wert zugeordnet (oder gleichgesetzt). COMP enthält hier den Punktestand des Computers und ist am Anfang des Spiels gleich Null.

40 Diese Zeile enthält die Variable SPI für den Punktestand des Spielers.

50 FOR... Damit fängt eine FOR/NEXT-Schleife an. Der Computer durchläuft eine solche Schleife so oft, wie es die letzte Zahl des FOR-Befehls zuläßt. Sehen Sie sich Zeile 280 weiter unten an (NEXT A). Hier ist das Ende der FOR/NEXT-Schleife. In diesem Programm muß der bedauernswerte Computer zehnmal hintereinander die gleiche Strecke von Zeile 50 bis Zeile 280 zurücklegen. Dabei führt er jedesmal alle im Programm enthaltenen Befehle aus.

60 CLS gibt, wie Sie schon wissen, den Befehl: Bildschirm löschen (s. o.). Der Doppelpunkt gibt Ihnen die Möglichkeit, dieser Zeile einen zweiten Befehl hinzuzufügen. Der PRINT-Befehl folgt als zweiter Befehl in dieser Zeile. Daraufhin werden die Wörter in den Anführungszeichen auf den Bildschirm geschrieben. Dazu schreibt der Computer den Wert von A. Der erste Durchgang durch die Schleife A entspricht der 1, der 2. Durchgang ist gleich 2 usw., bis A beim letzten Durchgang durch die Schleife gleich 10 ist. Mit dem Befehl PRINT „RUNDE NUMMER "; A wird auf dem Bildschirm „RUNDE NUMMER 4" angegeben, je nachdem, welchen Wert A gerade hat.

70–90 Diese drei Zeilen schreiben die Zahlen 1 bis 3 und das dazugehörige Wort (wie z. B. „Stein"). Beachten Sie in Zeile 70 das Apostroph (Taste 7). Das hat zur Folge, daß auf dem Bildschirm eine Leerzeile zwischen den Worten „RUNDE NR. 4" und „1 – STEIN" entsteht.

100 Das Apostroph wird wieder dazu benutzt, um vor dem Befehl „BITTE WÄHLEN: 1, 2 ODER 3" eine Leerzeile zu erhalten.

110 Der INPUT-Begriff wartet darauf, daß Sie eine Zahl eingeben. Bei diesem Spiel ist die Zahl, die Sie eingeben, der Variablen B zugeordnet.

120 Es erscheinen die Worte: „SIE WÄHLEN:"

130–150 Diese drei Zeilen interpretieren die eingegebene Zahl (1, 2, oder 3, jeweils B zugeordnet). Hier wird entschieden, welches Wort (Stein, Schere oder Papier) auf dem Bildschirm ausgegeben wird.

160 Durch den PAUSE-Befehl sieht es so aus, als ob der Computer mal kurz „nachdenkt". Die Zahl hinter dem Wort Pause gibt an, wieviel fünfzigstel Sekunde (in den USA sechzigstel Sekunde) die Pause dauert. PAUSE 50 bedeutet demnach, daß der kurze Stopp eine Sekunde andauert. Bei PAUSE 0 wartet der Spectrum eine unbegrenzte Zeit, bis Sie irgendeine Taste drücken.

170 Jetzt wird's interessant! Der Computer erzeugt in dieser Zeile eine Zufallszahl. Solche Zahlen sind in Spielprogrammen sehr nützlich. Die RND-Funktion (drücken Sie beide SHIFT-Tasten und dann die T-Taste) gibt Ihnen eine beliebige Zahl zwischen 0 und 1 an. Tippen Sie z. B. PRINT RND und danach die ENTER-Taste. Dann erhalten Sie eine Zahl zwischen 0 und 1. Zeile 170 wandelt das Ergebnis zu einer ganzen Zahl zwischen 1 und 3 ab. Falls die 3, die auf das eingeklammerte Sternchen folgt, z. B. in eine 10 umgewandelt würde, dann würde C eine vom Computer ausgewählte beliebige Zahl zwischen 1 und 10 zugeordnet werden. Bei diesem Spiel wollen wir, daß der Computer 1, 2, oder 3 wählt. Also multiplizieren wir RND mit 3. Mit INT erhalten wir den ganzzahligen Anteil, also 0, 1 oder 2, und addieren anschließend eine 1.

180 Diese Zeile sagt uns, was der Computer gewählt hat.

190–210 Diese Zeilen wandeln die Zahlen 1, 2 oder 3, die C zugeordnet sind, in die Wörter Stein, Schere und Papier um.

220 Die Variable D wird dem Wert 260 zugeordnet. Das hat zur Folge, daß der Computer am Ende der Zeilen 230 und 240 (Befehl: GO TO D) zur Zeile 260 weitergeht. Da der Spectrum ganz gerne rechnet, könnten Sie auch (obwohl es eigentlich unsinnig ist) die Zeilen 230 und 240 mit GO TO 2 * 130 (2 mal 130) enden lassen. Der Computer würde dann ebenfalls nach 260 gehen.

230 Wenn B die gleiche Zahl wie C ist, weiß der Computer, daß er das gleiche wie Sie gewählt hat. Er schreibt „Unentschieden" und rückt nach D, d. h. nach Zeile 260 vor (GO TO 260).

240 Diese Zeile sieht etwas kompliziert aus. Sie entscheidet darüber, ob sich aus bestimmten Kombinationsmöglichkeiten von B und C ein Punktgewinn für den Computer ergibt. Falls (IF) das der Fall ist, dann (THEN) schreibt der Computer (PRINT) „Ich gewinne" und erhöht seine Punktzahl um 1 (LET COMP = COMP +1). Der Befehl ist nicht so merkwürdig, wie er aussieht. Er bedeutet: Setze die Variable links vom Gleichheitszeichen im Wert dem Wert der alten Variablen plus 1 gleich. Wenn Sie das jetzt noch nicht verstehen sollten, keine Angst, es wird Ihnen nach und nach klar werden, wenn Sie solche Zeilen öfter benutzen.

250 Die Gewinnchancen des Spielers werden ermittelt. Gegebenenfalls erhöht sich seine Punktzahl um 1 (LET SPI = SPI +1).

260 Der Punktestand wird angezeigt. Dem PRINT-Befehl, von zwei Anführungszeichen eingerahmt, geht ein Apostroph voran. Zwei Apostrophe folgen. Daran schließt sich der ICH-Befehl mit einem Anführungszeichen an. Sie wissen jetzt schon vom bisherigen Programmablauf, daß diese Apostrophe Leerzeilen produzieren. Demzufolge wird die Zeile mit dem ICH-Befehl erst nach dem Einschub von zwei Leerzeilen ausgeschrieben. Das pfeilförmige Zeichen „größer als", das hinter den Wörtern ICH und SIE steht, bekommen Sie von der T-Taste, wenn Sie gleichzeitig die SYMBOL SHIFT-Taste herunterdrücken.

270 Während einer Pause von zwei Sekunden können Sie das Ergebnis dieser Runde lesen.

280 NEXT A signalisiert das Ende der FOR/NEXT-Schleife, die in Zeile 50 begann. Wie schon einmal erklärt, durchläuft der Computer auf diesen Befehl hin die Schleife von Zeile 50 an, wo der Wert von A um 1 erhöht wird, mit jedem einzelnen Folgebefehl aufs neue.

290 Wenn der Wert der Variablen SPI gleich dem Wert der Variablen COMP ist, dann weiß der Spectrum, daß das Spiel unentschieden ausgegangen ist. Er schreibt (nach zwei Leerzeilen, die mit zwei Apostrophen angezeigt werden): „Unentschieden".

300 Falls (IF) SPI kleiner als COMP ist, dann weiß der Computer, daß er gewonnen hat.

310 Falls (IF) COMP kleiner als SPI ist, dann weiß der Computer, daß Sie gewonnen haben.

Diese Erläuterungen waren hoffentlich nicht zu verwirrend. Aber es ist schwer, in kurzer Zeit alles Wesentliche über das Programmieren des Spectrum in BASIC möglichst verständlich zusammenzufassen.

Und jetzt – ein wenig Farbe

Später werden wir uns noch ausführlich überlegen, wie wir aus den Fähigkeiten des Spectrum, Farbe (und Ton) zu produzieren, das Beste machen können. Aber jetzt wollen wir erst einmal einige der einfachsten Möglichkeiten zeigen, Farb-Befehle zu benutzen. Sie werden erleben, wie wirkungsvoll Sie damit Programme bereichern können. Eine neue Version von STEIN, SCHERE, PAPIER mit zusätzlichen Farb-Befehlen folgt. Sie können das Programm leicht abändern, wenn Sie die Edit-Möglichkeiten des Spectrum verwenden.

```
  10 REM STEIN, SCHERE, PAPIER
  20 REM © HARTNELL, 1982
  30 LET comp=0
  40 LET spi=0
  50 FOR a=1 TO 10
  60 CLS: PRINT INK 2;"RUNDE NUM
MER "; INK 1;a
  70 PRINT INK 1;'"1 - STEIN"
  80 PRINT INK 2;"2 - SCHERE"
  90 PRINT INK 3;"3 - PAPIER"
 100 PRINT INK 6; PAPER 2;'"BITT
E WAEHLEN: 1,2 ODER 3!"
 110 INPUT b
 120 PRINT INK b;'"SIE WAEHLEN";
 130 IF b=1 THEN PRINT "STEIN"
 140 IF b=2 THEN PRINT "SCHERE"
 150 IF b=3 THEN PRINT "PAPIER"
 160 PAUSE 50
 170 LET c=INT (RND*3)+1
 180 PRINT INK c;'"ICH WAEHLE ";
 190 IF c=1 THEN PRINT "STEIN"
 200 IF c=2 THEN PRINT "SCHERE"
 210 IF c=3 THEN PRINT "PAPIER"
 220 LET d=260
 230 IF b=c THEN PRINT FLASH 1;'
"UNENTSCHIEDEN!": GO TO d
 240 IF c=1 AND b=2 OR c=2 AND b
=3 OR c=3 AND b=1 THEN PRINT FLA
SH 1; BRIGHT 1;'"ICH GEWINNE!":
LET comp=comp+1: GO TO d
 250 IF b=1 AND c=2 OR b=2 AND c
=3 OR b=3 AND c=1 THEN PRINT FLA
SH 1;'"SIE GEWINNEN!": LET spi=s
pi+1
 260 PRINT '"PUNKTE"'"ICH> ";co
mp;TAB 10;"SIE> ";spi
 270 PAUSE 100
 280 NEXT a
 285 INVERSE 1
 290 IF spi=comp THEN PRINT '"U
NENTSCHIEDEN!!!"
 300 IF spi<comp THEN PRINT '"I
CH HABE DIESES SPIEL GEWONNEN!!!
"
 310 IF comp<spi THEN PRINT '"S
IE HABEN DIESES SPIEL GEWONNEN!!
!"
```

Sie sehen, in Zeile 60 sind die Wörter INK 2 zum vorherigen Programm hinzugekommen. Die Wörter INK 2 bekommen Sie in die Zeile hinein, indem Sie erst LIST drücken (die K-Taste), dann 60 – jetzt haben Sie LIST 60 – und dann ENTER. Das Wort „scroll?“ erscheint unten auf

dem Bildschirm. Drücken Sie N, um weitere „scroll"-Fragen beim Anlisten zu vermeiden. Dann drücken Sie die CAPS SHIFT-Taste herunter und gleichzeitig die Taste 1, auf der oben EDIT steht. Die Zeile, auf die der Cursor (mit einem >) zeigt (in diesem Fall Linie 60), erscheint oben auf dem Bildschirm. Wir wollen jetzt die Wörter INK 2 hinter dem Wort PRINT einfügen: Drücken Sie die CAPS SHIFT-Taste, halten Sie sie fest und drücken Sie gleichzeitig die Taste 8. Sie sehen, wie der Cursor sich in Richtung des über der Taste 8 angegebenen Pfeils bewegt. Nehmen Sie erst wieder den Finger von der Taste 8, wenn der Cursor an PRINT vorbei ist! Weiter darf er nicht wandern! Drücken Sie jetzt beide Shift-Tasten zugleich und lassen Sie dann die CAPS SHIFT-Taste los. Die SYMBOL SHIFT-Taste müssen Sie weiter nach unten gedrückt halten. Drücken Sie jetzt die X-Taste, und wenn alles klappt, erscheint jetzt das Wort INK auf dem Bildschirm. Schließen Sie eine 2 und ein Semikolon an (SYMBOL SHIFT- und dann die O-Taste), und drücken Sie dann wieder ENTER. Sie sehen, wie die verbesserte neue Zeile den Platz einnimmt, den zuvor die ursprüngliche Zeile 60 innehatte.

Wenn Sie jetzt das Programm von neuem starten (RUN), dann sehen Sie die Wörter RUNDE NR. 1 in Rot auf dem Bildschirm erscheinen. Die Farben werden mit den Zahlen 0–7 gewählt (eine genaue Erklärung folgt später). Mit INK 1 werden die Buchstaben blau, mit INK 2 rot usw. Sie brauchen nicht das ganze Spiel durchlaufen zu lassen. Geben Sie einfach, wenn der Computer nach einer Zahl fragt, ein Q ein. Der Spectrum wird dann mit einer Fehlermeldung ganz verblüfft anhalten, weil er nicht weiß, was Q bedeutet.

Drücken Sie jetzt wieder EDIT (CAPS SHIFT- und 1-Taste), und von neuem erscheint die Zeile 60 unten auf dem Bildschirm. Mit der CAPS SHIFT/8-Kombination bewegen Sie den Cursor vor das A. Dann fügen Sie INK 1 und noch ein Semikolon hinzu. Dadurch wird die Verkettung der einzelnen Angaben im PRINT-Befehl gewährleistet. Bringen Sie mit ENTER diese Zeile wieder in ihre Position innerhalb des Programms und starten Sie es von neuem. Sie sehen jetzt die Wörter RUNDE NUMMER in Rot und die Zahl 1 in Blau auf dem Bildschirm.

Wenn die Farben nicht klar sind, spielen Sie ein wenig mit dem Kanalwähler und der Farbkontrolle am Fernseher, bis Sie Rot und Blau klar sehen können. Verschiedene Fernsehfabrikate geben die Farben unterschiedlich wieder, bei dem einen sind Sie intensiver als beim anderen.

Jetzt können Sie das ganze Programm durchgehen und die notwendigen Änderungen vornehmen, um überall Farbe hinzuzufügen. Die Zeilen 70, 80, 90, 100, 120, 180, 230, 240, 250 und 285 müssen abgeändert werden. 285 muß neu hinzugefügt werden, wird sich aber automatisch selbst einordnen. Starten Sie das Programm wieder (RUN). Durch die Farben wird das Programm wesentlich interessanter und lebendiger.

Anleitung zum Schreiben

Unser Abschlußprogramm in diesem Einführungskapitel soll Ihnen helfen, sich auf der Tastatur zurechtzufinden. Wenn Sie das Programm starten, wird auf dem Bildschirm eine beliebig gewählte Zahl oder ein Buchstabe aus dem Alphabet vorgegeben, und zwar in ungefähr der gleichen Position wie auf der Tastatur. Sie müssen innerhalb einer begrenzten Zeitspanne die jeweils bezeichnete Taste drücken. Wenn es nicht klappt, erlaubt Ihnen der verständnisvolle Computer noch einen Versuch. Sie werden sehen, wie schnell Sie mit Hilfe dieser Routine den Aufbau der Tastatur beherrschen lernen.

Wenn Sie mehr Zeit brauchen, um jeweils die gewünschte Taste zu finden, dann ersetzen Sie die 100 in Zeile 180 durch eine größere Zahl. Fangen Sie mit 300 an und gehen Sie dann langsam herunter, wenn Sie sich die Tippgeschwindigkeit zutrauen. Versuchen Sie, bei der Eingabe der Zahlen und Buchstaben in den DATA-Befehlen (in den Zeilen 80–110) keinen Fehler zu machen. Denn damit geben Sie an, an welcher Stelle auf dem Bildschirm die verschiedenen Zahlen und Buchstaben geschrieben werden. Nachdem Sie das Programm ein paar Mal ausgeführt haben, gehen Sie bitte anschließend die dazugehörige Erklärung im Buch Zeile für Zeile durch.

```
  10 REM SCHREIBUEBUNGEN
  20 DIM a$(36,5)
  30 LET punkte=0
  40 FOR a=1 TO 36
  50 READ b$
  60 LET a$(a)=b$
  70 NEXT a
  80 DATA "10502","20605","30606
","40611","50614","60617","70620
","80628","90626","00629"
  90 DATA "A1202","B1517","C1511
","D1208","E0908","F1211","G1214
","H1217","I0923","J1220"
 100 DATA "K1223","L1226","M1525
","N1520","O0926","P0929","Q0902
","R0911","S1205","T0914"
 110 DATA "U0920","V1514","W0905
","X1508","Y0917","Z1505"
 120 REM
 130 REM    DER TEST
 140 REM
 150 FOR a=1 TO 10: BEEP .02,-10
: CLS
 155 IF INKEY$<>"" THEN GO TO 15
5
 160 LET b=INT (RND*36)+1
 170 PRINT FLASH 1; PAPER 6;AT V
AL (a$(b)(2 TO 3)),VAL (a$(b)(4
TO ));a$(b,1)
 180 FOR c=1 TO 100
 190 LET c$=INKEY$
 200 IF c$<>"" THEN GO TO 260
```

```
 210 NEXT c
 220 PRINT AT 0,0; INK 7; PAPER
4; FLASH 1; BRIGHT 1;"SIE BRAUCH
TEN ZU LANGE!"
 225 FOR d=1 TO 70: NEXT d
 230 NEXT a
 240 GOTO 300
 260 IF c$=a$(b,1) THEN LET punk
te=punkte+1: BEEP .25,punkte*4:
PRINT AT 16,0; PAPER 6; INK 2;"G
UT GEMACHT. IHRE PUNKTZAHL IST N
UN ";punkte
 265 IF c$<>a$(b,1) THEN PRINT A
T 0,0; INK 1; PAPER 6; FLASH 1;"
DAS WAR LEIDER FALSCH!";TAB 10;"
BITTE NOCHMAL!": GO TO 170
 270 FOR d=1 TO 100
 280 NEXT d
 290 NEXT a
 300 CLS : PRINT AT 8,2; FLASH 1
; BRIGHT 1; PAPER 6; INK 2;"IHRE
 PUNKTZAHL WAR ";punkte;" VON 10
"
 310 IF punkte>7 THEN PRINT AT 1
0,7; FLASH 1; INK 4; PAPER 7; BR
IGHT 1;"HERZLICHEN GLUECKWUNSCH!
"
```

10 Dies ist ein REM-Befehl (REM bedeutet „Bemerkung" oder „Überschrift", wie Sie sich erinnern), den der Computer ignoriert. Sie können dabei beliebig viele Leerstellen verwenden.

20 Zeile 20 legt die Größe eines Feldes fest. Mit einem Feld können Sie Zahlen oder Texte hintereinander speichern und bei Bedarf abrufen, indem Sie seine Position im Feld angeben. Das in Zeile 20 definierte Feld heißt A$. Das Dollarzeichen bedeutet, daß in dem Feld keine Zahlen, sondern Texte gespeichert werden sollen. Im Programm erkennen Sie Texte daran, daß Sie in Anführungszeichen eingerahmt werden.

30 Die Variable PUNKTE notiert, wieviel Buchstaben Sie richtig tippen.

40– Dies ist eine FOR/NEXT-Schleife wie im vorigen Programm.
70 Zeile 50 (READ B$) steht in Verbindung mit den DATA-Befehlen 80–110. Der Reihe nach werden alle DATA-Angaben vom Programm gelesen. Beim ersten Durchlauf der Schleife wird die erste Zahl in der DATA-Zeile („10602") eingelesen und B$ zugeordnet. Zeile 60 setzt dann A$(1) auf den Wert von B$. Bei der nächsten Schleife, wenn A gleich 2 ist, liest sich die Zeile 60 wie folgt: LET A$(2) = B$. Diesmal hat B$ den Wert der zweiten Zahl des DATA-Befehls („20605"). Und so geht es weiter, bis alle 36 Plätze der A$-Liste einen Wert aus den Data-Zeichen erhalten haben.

80–110 Diese Zeilen enthalten die Data-Angaben. Beachten Sie bitte: Jeder Wert ist mit Anführungszeichen versehen und durch ein Komma vom nächsten getrennt.

120–140 Diese Zeilen enthalten drei REM-Befehle.

150 Hier beginnt eine neue FOR/NEXT-Schleife. Solche Schleifen müssen nicht immer A genannt werden. Sie können jeden Buchstaben des Alphabets verwenden, z. B. FOR I = 1 TO 10 oder FOR M = 1 TO 10. Bei BEEP .02,–10 erklingt ein kurzer Ton (BEEP wird später genauer erklärt), und der CLS-Befehl löscht den Bildschirm.

155 INKEY$, das Sie oben auf der N-Taste finden, liest die Tastatur und wartet so lange, bis Sie die Tastatur losgelassen haben (Achtung: das <>-Zeichen befindet sich auf der W-Taste).

160 B enthält hier einen Wert zwischen 1 und 36, um eine Zahl zwischen 0 und 9 oder einen Buchstaben des Alphabets zu wählen.

170 Dies ist eine ziemlich komplizierte Zeile. Sie schreibt den Buchstaben oder die Zahl, die an der ersten Stelle des jeweiligen A$-Wertes stehen, mit A$(B,1) genau dorthin, wo sie hingehören: Hierzu wird der PRINT AT-Befehl verwendet, mit dessen Hilfe die genaue Bildschirmposition festgelegt werden kann. Ziffer 2 und 3 des jeweiligen DATA-Wertes werden benutzt, um die Bildschirmzeile festzulegen (beginnend mit Null). Ziffer 4 und 5 werden benutzt, um den Abstand vom linken Rand (d. h. die Spalte) festzulegen. Mit anderen Worten, jeder DATA-Wert enthält einen Buchstaben oder eine Zahl als erste Angabe. Deren Plazierung auf dem Bildschirm wird durch die folgenden vier Ziffern angegeben. Die Plazierungsangaben 2 und 3 bestimmen die senkrechte Position, die Plazierungsangaben 4 und 5 die waagerechte Position.

180–210 Diese FOR/NEXT-Schleife wartet eine begrenzte Zeit auf einen Tastendruck. Die Länge der Pause wird mit der Zahl am Ende der Zeile 180 angegeben. Wenn in der Zwischenzeit eine Taste gedrückt wurde, gibt die Zeile 200 den Befehl zum Weitermachen an Zeile 260.

220 Wenn Sie nicht rechtzeitig eine Taste drücken, meldet der Computer „SIE BRAUCHEN ZU LANGE".

225 Hier gibt's eine kurze Pause mit einer FOR/NEXT-Schleife. Sie können dafür die PAUSE-Taste oder, wie hier gezeigt, eine „leere" FOR/NEXT-Schleife benutzen.

230 Der Computer geht zurück zur Zeile 150 zur nächsten Übung.

240 Nach zehn Durchläufen geht der Computer zur Zeile 300 und gibt den Punktestand an.

260 Hier wird geprüft, ob die gedrückte Taste die gleiche ist wie die, die auf dem Bildschirm zu sehen ist. Wenn ja, dann wird ein Punkt dem Punktestand (SCORE) hinzugefügt und Sie sehen die Meldung: „GUT GEMACHT, IHRE PUNKTEZAHL IST NUN 3" oder entsprechend.

265 Wenn die von Ihnen gedrückte Taste nicht mit dem Zeichen auf dem Bildschirm übereinstimmt, dann steht dort: „DAS WAR LEIDER FALSCH! BITTE NOCHMAL". Und zurück geht's, damit Sie einen neuen Versuch wagen können. Beachten Sie bitte, daß *hinter* dem Anführungszeichen nach den Wörtern „BITTE NOCHMAL" ein Komma steht. Lassen Sie es einmal weg und passen Sie auf, was sich jetzt dadurch ändert.

270–280 Hier gibt es eine Pause, bevor der nächste Buchstabe oder die nächste Zahl auf dem Bildschirm erscheint.

290 Das Ende der FOR/NEXT-Schleife ist erreicht. Das Programm geht zurück zur Zeile 150.

300 Der Punktestand wird angegeben.

310 Wenn mehr als sieben Punkte erreicht sind, gratuliert der Computer.

Dies ist wirklich ein sehr nützliches Programm, um mit der Tastatur vertraut zu werden, finden Sie nicht auch?

Der Inhalt dieses Kapitels wurde etwas komprimiert vorgelegt. Aber wenn Sie die einzelnen Abschnitte und Programmteile gründlich durchgegangen sind, müßten Sie jetzt schon einen guten Einstieg in das Programmieren mit BASIC auf dem Spectrum gefunden haben.

Die vorliegenden Texte sind im Prinzip als Ergänzungen zu den mitgelieferten Handbüchern zu verstehen. Im folgenden gehen wir davon aus, daß Sie die Grundbegriffe dieses Kapitels beherrschen und auch Kenntnisse aus den ZX Spectrum-Handbüchern erworben haben. Wenn Sie Bedenken haben, sollten Sie an dieser Stelle vielleicht noch einmal einen Blick in die Handbücher werfen und sich den einen oder anderen BASIC-Begriff in Erinnerung rufen. Die Erläuterungen zu den einzelnen Programmen sind jetzt nicht mehr ganz so ausführlich wie in diesem Kapitel. Aber vieles wird schon beim Eintippen der Programmzeilen oder bei der Ausführung des Programms unmittelbar klar werden.

Kapitel 3

Experimente mit den Farben des ZX Spectrum

Der Spectrum hat sechs Farben (dunkelblau, rot, violett, grün, hellblau und gelb) und zusätzlich schwarz und weiß. Jeder Farbe ist eine Nummer zwischen 0 und 7 zugeordnet:

0 Schwarz
1 Dunkelblau
2 Rot
3 Violett
4 Grün
5 Hellblau
6 Gelb
7 Weiß

Wenn Sie einen Schwarzweiß-Fernseher haben oder die Farbkontrolle an Ihrem Farbfernseher ganz herunterstellen, sehen Sie diese Farben in der vorgenannten Reihenfolge in abgestuften Grautönen, die von ganz dunkel (schwarz) bis ganz Hell (weiß) reichen. Versuchen Sie einmal folgendes:

```
10  FOR X = 0 TO 7
20  PRINT INK X; "■■" (zwei ausgefüllte Quadrate, die Sie bekommen,
                       indem Sie die CAPS SHIFT-Taste und dann die
                       Taste 9 und die Taste 8 drücken)
30  NEXT X
```

Die Farbbefehle geben auch an, wie die Farben auf dem Fernseher produziert werden. Ein Farbfernseher benutzt die Farben blau, grün und rot. Alle anderen Farben können aus diesen drei Grundfarben gemischt werden. Violett entsteht aus Blau und Rot, und die ihr zugeordnete Zahl 3 ist die Summe der beiden anderen Farbzahlen 1 und 2.

Beim Einschalten des Spectrum sehen Sie, daß der Bildschirm weiß ist. Sie schreiben darauf mit Schwarz. Mit dem BORDER-Befehl auf der B-Taste bringen Sie auf einfache Weise Farbe ins Bild. Versuchen Sie es:

```
BORDER 1 (anschließend ENTER)
```

Der Bildschirmrand wird jetzt dunkelblau. So können Sie dem Bildschirmrand jede der acht zur Verfügung stehenden Farben geben, wenn Sie nach dem BORDER-Befehl eine der Farbtasten und anschließend ENTER drücken. Auch für das Bildinnere und die Schrift kann Farbe verwendet werden. Bei PAPER geben Sie dem Viereck innerhalb des Bildschirmrandes Farbe und bei INK dem Text und den graphischen Zeichen. Die PAPER- und INK-Befehle sind etwas schwieriger als der BORDER-Befehl. Mit ihnen werden die jeweiligen Farben für den Hintergrund (PAPER) und für die Schrift (INK) festgelegt. Um die Farbe des Hintergrundes zu verändern, drücken Sie erst die weiße Shift-Taste auf der rechten Seite. Während Sie noch die rote Shift-Taste unten halten, drücken Sie die C-Taste. Probieren Sie's mal (beachten Sie bitte bei den folgenden Beispielen: Genauso wie beim BORDER-Befehl oben lassen wir auch hier die Zeilennummern weg, damit der Computer die Befehle sofort ausführt!):

```
PAPER 3:CLS (der CLS-Befehl ist auf der V-Taste)
PAPER 2:CLS
PAPER 6:CLS
```

Sie sehen: Alle Stellen auf dem Bildschirm werden erst violett, dann rot und dann gelb. Wie Sie wahrscheinlich wissen, können Sie jede der acht Farb-Zahlen mit dem PAPER-Befehl eingeben. Um mit der Farbe das ganze Viereck auszufüllen, müssen Sie dem PAPER-Befehl einen Lösch-Befehl (CLS) folgen lassen. Hierdurch wird an allen Bildschirmstellen ein Leerzeichen ausgegeben mit der entsprechenden Hintergrundfarbe.

Der INK-Befehl läßt sich genauso einfach verwenden. Drücken Sie wieder die CAPS SHIFT-Taste und dazu die SYMBOL SHIFT-Taste auf der rechten Seite. Während Sie noch die rote Shift-Taste festhalten, drücken Sie nun die x-Taste. Der INK-Befehl ist also ähnlich wie der PAPER-Befehl zu handhaben.

Versuchen Sie einmal folgendes, um die Wirkung von farbigem Text zu erleben:

```
10 PAPER 7
20 INK 3
30 CLS
40 PRINT "Test"
```

Wenn Sie das Programm starten, sehen Sie, wie der Computer den violetten Text auf eine weiße Fläche schreibt. Sie können zwar jeden farbigen Text auf jedes farbige Bildschirmviereck schreiben, aber manche Farben passen schlecht zueinander.

Das sehen Sie am besten, wenn Sie beispielsweise das folgende Programm laufen lassen:

```
10 PAPER 5
20 INK 4
30 CLS
40 PRINT "Test"
```

Jetzt können Sie die Wörter nur schwer erkennen. Rot und Violett ergänzen sich nicht gut, und violette Farbe auf grüner Fläche ist ganz besonders schwer zu lesen.

Sie können mit den Farb-Befehlen den Bildschirm auch nur teilweise statt flächendeckend ausfüllen. Versuchen Sie folgendes, um das einmal zu sehen:

```
PAPER 7: CLS: PRINT INK 5; PAPER 1; "Test"
```

Jetzt ist nur die Fläche hinter den Wörtern hellblau, während der Text dunkelblau ist.

Die Farbe kann auf viererlei Art verändert werden. Die BASIC-Wörter dafür finden Sie unter der unteren Tastenreihe Ihres Spectrum: FLASH, BRIGHT, OVER und INVERSE (blinken, hell/dunkel, überschreiben und umgekehrt darstellen). Die beiden letzten Wörter sind ausgesprochen vielseitig nutzbar beim Umgang mit Farbe, aber sie werden auch in anderen Bereichen verwendet.

BRIGHT

Der BRIGHT-Befehl ist leicht zu verstehen. Mit ihm wird die Farbe des Textes (INK) oder der Fläche (PAPER) etwas hervorgehoben. Die Farben werden leuchtender und manchmal sogar einen Ton heller. Folgende Routine zeigt die Wirkung des BRIGHT-Befehls:

```
10 FOR x=0 TO 7
20 PRINT PAPER x; BRIGHT 0; " "; BRIGHT 1; " ";
30 NEXT x
```

Damit werden Leerstellen in Zweierblöcken ausgegeben (denken Sie dabei an kleine Schreibblöcke). Der erste hat die normale Farbe, der zweite hat die gleiche Farbe, ist aber deutlich heller. Wie Sie sehen, folgt dem

BRIGHT-Befehl eine Null bzw. eine Eins. Mit Null wird die Farbe normal, mit Eins wird sie leuchtend wiedergegeben.

Das nächste Programm kommt zu einem ähnlichen Ergebnis, indem es die BRIGHT-Kontrolle in eine Schleife steckt:

```
10  FOR x = 0 TO 7
20  FOR y = 0 TO 1
30  PRINT INK x; BRIGHT y;" ";
40  NEXT y
50  NEXT x
```

FLASH

Der nächste Begriff heißt FLASH. Er wird so ähnlich wie BRIGHT benutzt und wird ebenso von einer Null oder einer Eins gefolgt, je nachdem, ob das Blinken aus- oder eingeschaltet ist. Mit der folgenden Routine können wir die leuchtenden BRIGHT-Quadrate des vorigen Beispiels z. B. aufblitzen (FLASH) lassen:

```
10  FOR x = 0 TO 7
20  FOR y = 0 TO 1
30  PRINT INK x; BRIGHT y; FLASH y; " ";
40  NEXT y: NEXT x
```

Die Routine zeigt, daß FLASH ständig zwischen der Farbe von INK und der Farbe von PAPER wechselt.

Eine Besonderheit stellt der Befehl INK 9 dar. Dieser Wert wird beim Einschalten des Rechners automatisch als erstes verwendet. Er besagt, daß alle Schriften jeweils in der bestmöglichen Kontrastfarbe zum Hintergrund dargestellt werden. Dies bedeutet bei hellem Hintergrund, daß die Schrift in Schwarz erscheint, und bei dunklem Hintergrund (PAPER-Farben 0–3), daß die Schrift in Weiß (INK 7) erscheint. Selbstverständlich können Sie diesen Befehl auch im Programm benutzen.

Logischerweise gibt es auch die Schriftfarbe 8. Dies bedeutet, daß der Spectrum an einer aktuell anzugebenden Stelle erst nachschaut, welche Farbe das bisherige Zeichen an dieser Stelle hatte, um dann das neue Zeichen mit der gleichen Farbe zu bringen.

Die gleichen Bemerkungen gelten entsprechend für die Befehle PAPER 8 bzw. PAPER 9 zur Festlegung der bisherigen bzw. der optimalen Hintergrundfarbe.

Mit FLASH 8 bzw. BRIGHT 8 wird auf die früher für eine bestimmte Bildschirmstelle festgelegten Farbeigenschaften zurückgegriffen. Lediglich FLASH 9 bzw. BRIGHT 9 ergeben keinen Sinn: Blinken kann nicht noch wirkungsvoller werden, und auch die Helligkeit kann nicht noch mehr verstärkt werden.

INVERSE, OVER

Die beiden Begriffe, die für den Umgang mit Farbe übrigbleiben, lauten: INVERSE und OVER. INVERSE steckt im Prinzip hinter dem Befehl FLASH, nämlich die Farbe des Textes in die Farbe der Hintergrundfläche zu verwandeln und umgekehrt. Dadurch wird die weiße Schrift auf schwarzem Grund zu Schwarz auf Weiß und umgekehrt.

Das klappt bei allen Farben für INK oder PAPER und hat eine ähnliche Wirkung wie BRIGHT und FLASH. Der Befehl INVERSE wird von Null (für „aus") oder EINS (für „an") gefolgt. Allerdings können Sie nicht die 8 auf dieses Wort folgen lassen.

Versuchen Sie folgendes, um zu sehen, wie es funktioniert:

```
10 FOR x = 0 TO 1
20 PRINT INVERSE x; INK 5; PAPER 1;
   "Dies zeigt, wie INVERSE arbeitet"
30 PAUSE 50
40 NEXT x
```

OVER ist ein sehr nützlicher BASIC-Begriff. Damit können Sie ein Zeichen mit einem anderen überlappen. Auch hier gilt Null für „aus" und Eins für „an". Wie bei INVERSE können Sie auch nach OVER nicht die 8 benutzen. OVER trifft bei einem Zeichen-Viereck eine Entweder/Oder-Entscheidung. Das heißt, es überlagert nicht nur ein Zeichen durch ein anderes. Das Ergebnis dieses Befehls ist, daß alle Punkte des Zeichenvierecks verschwinden, die bei der Überlappung in beiden Punkterastern vorkommen.

Sie können am besten verstehen, wie OVER funktioniert, wenn Sie bedenken, daß jedes Zeichen-Quadrat (es gibt 32 waagerechte und 24 senkrechte) aus 64 winzigen Punkten auf einer 8 x 8 Matrix besteht (ungefähr so wie ein Schachbrett oder kariertes Zeichenpapier). Wenn der OVER-Befehl vom Computer verlangt, über etwas hinwegzuschreiben, dann schaut sich der Spectrum den gleichen Punkt auf den beiden in Frage kommenden Zeichen-Quadraten an. Mit Hilfe der „Wahrheitstafel" entscheidet er dann, ob der Punkt schließlich in Schwarz oder Weiß ausgeführt wird:

altes Zeichen \ neues Zeichen	schwarzer Punkt	weißer Punkt
schwarzer Punkt	weiß	schwarz
weißer Punkt	schwarz	weiß

Nehmen wir ein Beispiel, um das noch klarer zu machen! Schreiben Sie zwei Zeichen auf Ihren Bildschirm, den Großbuchstaben O und die Null 0:

```
PRINT "O0"
```

Stellen Sie sich jetzt das Resultat vor, wenn Sie den Großbuchstaben „O" über die Null schreiben (wobei Sie daran denken, daß zwei schwarze Punkte einen weißen ergeben), und versuchen sie jetzt, den OVER-Befehl in der Praxis auszuführen:

```
PRINT OVER 1;"O";CHR$8;"0"
```
(Beachten Sie, daß CHR$8 ein Kontrollzeichen zum Zurücksetzen ist.)

Auf dem Bildschirm erscheint ein Schrägstrich aus dem Innern der Null. Der Rest wurde bei der Überlagerung ausgelöscht. Versuchen Sie jetzt, einige andere Zeichen mit OVER zu überschreiben und vorherzusagen, was dann passiert.

Wie können Sie den OVER-Befehl noch verwenden? Eine interessante Möglichkeit wäre, Objekte vor oder hinter anderen vorbeiwandern zu lassen. Ein einfaches Beispiel dafür gebe ich im Programm ÜBERLAGERUNG (s. u.). Darin wandert ein normaler Buchstabe p über den Bildschirm zu einem inversen p. Das normale p wandert über das andere und erzeugt im gleichen Augenblick ein schwarzes Quadrat (erinnern Sie sich, daß schwarz und weiß schwarz erzeugt?). Dann wandert es weiter, während das inverse p unverändert stehen bleibt.

```
   5 PAPER 5: INK 1: CLS
  10 PRINT INVERSE 1;AT 10,10;"p
"
  15 FOR a=0 TO 20
  20 PRINT PAPER 1; INK 5; INVER
SE 1; OVER 1;AT 10,a;"p"
  30 PAUSE 10
  40 PRINT INVERSE 1; OVER 1;AT
10,a;"p"
  50 NEXT a
```

ATTR

Wie Sie schon wissen, werden pro Bildschirmspeicherstelle nur je eine Information für die Schriftfarbe und eine Information für die Hintergrundfarbe notiert. Der Spectrum speichert nämlich die INK-, PAPER-, FLASH- und BRIGHT-Informationen einer jeden Position in einem besonderen Speicherbereich für „Attribute" (von Platz 22528 bis Platz 23295). Dies ergibt 768 Speicherstellen, nämlich 32 x 24, d. h. die Anzahl der Spalten multipliziert mit der Anzahl der Reihen auf dem Bildschirm. Jede dieser Speicherstellen enthält eine Zahl zwischen 0 und 255, die besagt, welche Farben vorhanden sind usw. Diese Zahl können Sie sich anschauen, wenn Sie den Sinclair-BASIC-Begriff ATTR (x,y) benutzen. x und y geben wie in PRINT AT die Koordinaten jedes in Frage kommenden Quadrates an. ATTR liefert eine Zahl, deren Bedeutung Sie nach einiger Übung sehr schnell interpretieren können. Diese Zahl setzt sich aus der Summe der folgenden Zahlen zusammen:

128 oder 0	je nachdem ob das Quadrat aufblinken soll (FLASH)
64 oder 0	je nachdem ob es leuchten soll (BRIGHT) oder nicht
8	mal dem Farbcode für PAPER
	und schließlich dem Code für INK

Wir können ATTR wie folgt benutzen:

```
PRINT AT 10,10; FLASH 1; BRIGHT 1; INK 2;
PAPER 6; "S"; ATTR (10,10)
```

Hier schreiben wir den Buchstaben S in Zeile 10, Spalte 10 mit roter Schrift auf gelber Fläche und geben noch zusätzlich den BRIGHT- und FLASH-Befehl. Was, glauben Sie, wird der Computer als ATTR der Position 10,10 angeben? Es müßte 128+64+8*6+2= 242 sein. Das ist ziemlich einfach, vorausgesetzt, daß Sie die Bestandteile kennen. Wenn Sie die Antwort auf ATTR (x,y) haben und die Bedeutung der Zahl herausbekommen wollen, dann benutzen Sie das Attribut-Berechnungs-Programm (s. u.).

Sie könnten in einem Programm die Variable A mit dem Ergebnis von ATTR (x,y) gleichsetzen und die erhaltenen Werte für INK, PAPER, BRIGHT und FLASH benutzen. ATTR kann gut dazu verwendet werden, die Eigenschaften eines Zeichens in einer bestimmten Position zu ermitteln. Vielleicht möchten Sie beispielsweise mit einem Raumschiff fahren, bis es an einen Berg stößt. Sie können dann den Berg in einer anderen Farbe zeichnen als den Rest des Bildes. Indem Sie testen, ob vor dem Raumschiff noch Platz für ein Zeichen ist oder ob dieser schon vom Berg besetzt ist, können Sie voraussagen, wann der Zusammenstoß passiert.

BERECHNUNG DER ATTRIBUTE

```
   10 INPUT "ATTRIBUTWERT?";a
   20 IF a-128<0 THEN GO TO 50
   30 LET a=a-128: PRINT "FLASH A
N,";
   40 GO TO 60
   50 PRINT "FLASH AUS,",
   60 IF a-64<0 THEN GO TO 90
   70 LET a=a-64: PRINT "BRIGHT A
N,",
   80 GO TO 100
   90 PRINT "BRIGHT AUS,",
  100 IF a/8=INT (a/8) THEN GO TO
  140
  120 PRINT "PAPER IST ";INT (a/8
),",INK IST ";a-(8*INT (a/8))
  130 STOP
  140 PRINT "INK IST 0, PAPER IST
  ";a/8
```

In diesem Fall wissen wir, wo sich der Berg befindet. Der Abruf von ATTR wäre hier also ziemlich witzlos. Wir könnten uns jedoch Planeten vorstellen, die ständig in Bewegung sind, so daß wir nicht genau wissen, wo wir auf ein Hindernis stoßen, oder dessen Position erst durch viele IF...THEN-Befehle ermitteln müßten. Dann ist es sinnvoll, ATTR zu benutzen.

Wie ein Regenbogen gezeichnet wird

Jetzt versuchen wir einmal, in hochauflösender Grafik einen farbigen Regenbogen zu zeichnen. Wir malen farbige Kurven, indem wir den Befehl,

```
    1 REM REGENBOGEN
    2 LET x=1
    3 DIM a(6): GO SUB 100
    4 LET y=1
    5 PAPER 7: BORDER 0: CLS
   10 FOR a=160 TO 250 STEP 15
   20 FOR u=1 TO 5
   30 INK a(y): PLOT a+u,0: DRAW
-a,a-80+u,PI/2
   40 NEXT u
   50 LET y=y+1
   60 IF y=7 THEN GO TO 200
   70 NEXT a
  100 LET a(1)=3: LET a(2)=1: LET
a(3)=5: LET a(4)=4: LET a(5)=6:
LET a(6)=2
  110 RETURN
  200 PRINT PAPER 7;AT 19,5; INK
3;"S"; INK 1;"P"; INK 5;"E"; INK
 4;"CT"; INK 6;"R"; INK 2;"UM"
```

Kreisbögen zu zeichnen, etwas abändern. Dabei ist jeder Kreisbogen vom nächsten, andersfarbigen Kreisbogen genau ein Leerzeichen entfernt. Wissen Sie noch, warum das so ist? Pro Bildschirmstelle gibt es nur eine INK-Angabe. Wenn also zwei verschiedenfarbige Linien zu nahe aneinander kommen, kann an den Berührungsstellen im Bereich eines Quadrates nur die Farbe der einen Linie oder die Farbe der anderen Linie gelten, aber nicht beides gleichzeitig. Denn jedes Quadrat kann nur eine, nicht aber zwei Farben aufnehmen.

128 Farben

Die Möglichkeit, zwei Farben (eine für die Schrift, eine für den Hintergrund) pro Bildschirmspeicherstelle unterzubringen, können wir nutzen, um bis zu 128 Farben auf unserem Spectrum zu produzieren. Dazu müssen wir zunächst ein Grafikzeichen erstellen, welches so ähnlich wie ein kleines Schachbrett 8 x 8 aussieht (wir erinnern uns: ein solches Zeichen gibt es auf der A-Taste des ZX81).

Indem Sie den PAPER-Befehl in einer Farbe und den INK-Befehl in einer anderen Farbe festlegen, nimmt das Zeichen halb die eine und halb

```
  10 REM FARBEN
  20 PAPER 7: BORDER 6: CLS
  30 REM SCHACHBRETT - ZEICHEN
  40 FOR x=0 TO 6 STEP 2
  50 POKE USR "P"+x,85
  60 POKE USR "P"+x+1,170
  70 NEXT x
  80 FOR p=0 TO 7
  90 FOR i=0 TO 7
 100 FOR b=0 TO 1
 110 PRINT PAPER p; INK i; BRIGH
T b;"PP";
 120 NEXT b: NEXT i: NEXT p
 130 PAUSE 200
 140 CLS : PRINT "UND NUN EIN VO
LLER BILDSCHIRM..": PAUSE 200
 150 CLS
 160 POKE 23692,100
 170 FOR p=0 TO 7
 180 FOR i=0 TO 7
 190 FOR b=0 TO 1
 200 FOR k=0 TO 31
 210 PRINT PAPER p; INK i; BRIGH
T b;"PPPPPPPPPPPPPPPPPPPPPPPPPPP
PPPPP"
 220 REM ALLE P'S SIND GRAFIK-ZE
ICHEN MIT DER TASTE 'P'
 230 NEXT k
 240 POKE 23692,100:
    REM AUTOMATISCHES ROLLEN
 250 NEXT b: NEXT i: NEXT p
 260 STOP
```

die andere Farbe an. Die beiden Farben stoßen unmittelbar aufeinander und vermischen sich deshalb. Rote und gelbe Farben ergeben z. B. ein schönes Orange. Probieren Sie einmal aus, wie viele verschiedene Farben Sie mit dem folgenden Programm auf Ihrem Spectrum produzieren können.

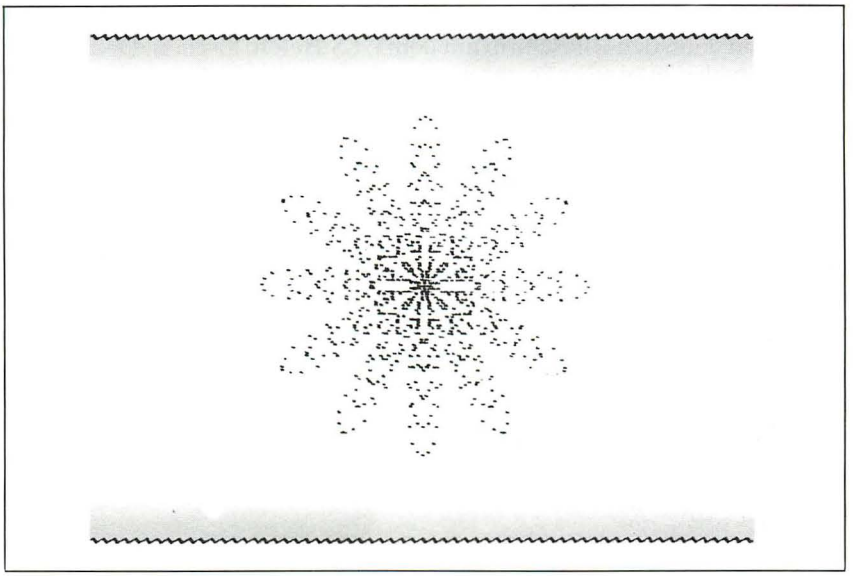
Wir erhalten das gleiche Ergebnis, wenn wir mit DRAW- und PLOT-Befehlen kreuzweise Linien auf dem Bildschirm zeichnen. Dabei müssen die Linien genau einen Bildpunkt voneinander entfernt verlaufen. Zeichnen Sie jetzt mit einer INK-Farbe auf einem andersfarbigen Hintergrund. Die Wirkung ist die gleiche wie bei dem graphischen Zeichen im vorigen Programm.

Der Spectrum zeigt sehr gut, was er alles kann, wenn Sie ihn mit dem DRAW-Befehl eine Art „durchbrochenes Muster" zeichnen lassen, wie das z. B. im Programm NETZWERK geschieht.

Hierbei werden vom Mittelpunkt des Bildschirms immer wieder neue Linien zu den Rändern des Bildschirms gezeichnet. Wenn Sie zu jeder Ecke des Bildschirms Verbindungslinien ziehen, bekommen Sie schließlich einen Farbklumpen. Wenn Sie aber winzige Punkte zeichnen, die immer eine Punktbreite voneinander entfernt sind, dann bekommen Sie ein sehr interessantes Muster. Das liegt daran, daß der Spectrum keine ganz geraden Linien zeichnen kann. Je nachdem, welcher Punkt und von welchem Punkt aus gezeichnet wird, unterscheiden sich die Linien beträchtlich voneinander. Das folgende Programm übermalt jede nur mögliche PAPER-Farbe mit jeder INK-Farbe. Das jeweilige Ergebnis hängt von den Farbmischungen ab. Manche Farben überlagern sich sehr gut und bilden farbige Strudel. Dieser optische Eindruck entsteht durch die Art der Überlagerung.

```
  10 REM NETZWERK
  20 LET z=1
  30 LET tinte=(RND*5)+1
  40 LET papier=(RND*5)+1
  50 IF tinte=papier THEN GO TO
30
  55 PAPER papier: INK tinte: CL
S
  60 FOR a=1 TO 2
  70 FOR x=0 TO 254 STEP 2
  80 PLOT 128,88: DRAW (-127*z)+
(x*z),z*-87
  90 NEXT x
 100 FOR y=0 TO 175 STEP 2
 110 PLOT 128,88: DRAW 127*z,z*-
87+(y*z)
 120 NEXT y
 130 LET z=-z
 140 NEXT a: PAUSE 100: GO TO 20
```

Das folgende Programm ist mindestens ebenso interessant wie NETZWERK. Es heißt EISBLUME und zeigt zunächst einen weißen Stern mit



```
  10 REM EISBLUME
  20 INK 7: BORDER 3: PAPER 1: C
LS
  30 LET p$=" ": LET t$=" "
  40 FOR x=0 TO 702
  50 LET p$=p$+t$
  60 PRINT ".";: NEXT x
  70 REM EIN STRING IN DER GROES
SE DES BILDSCHIRMS WIRD HIER ANG
ELEGT
  80 CLS
  90 LET schalter=0
 100 FOR k=20 TO 80 STEP 10
 110 LET y=0: LET z=PI
 120 LET c=100
 130 LET e=(z-y)/c
 140 FOR l=y TO z STEP e
 150 LET s=k*COS (l*6)
 160 LET h=s*SIN l
 170 LET v=s*COS l
 180 PLOT 128+h,88+v
 190 NEXT l
 200 LET schalter=schalter+1
 210 IF schalter=101 THEN GO TO
230
 220 GO TO 150
 230 LET schalter=0: NEXT k
 240 LET farbe=INT (RND*6)+1
 250 PRINT AT 0,0;PAPER farbe; O
VER 1;p$
 260 PAUSE 100: GO TO 240
```

vielen „Zacken". Bei der folgenden Technik lernen Sie eine neue Seite Ihres Spectrum kennen: Über die Schneeflocke wird eine neue PAPER-Farbe gelegt. Sie erinnern sich, wie das normalerweise geht: Um die Farbe der Bildschirmfläche zu ändern, müßten Sie erst die Farbnummer angeben und dann den Bildschirm mit dem CLS-Befehl löschen. Damit würde aber auch das von Ihnen gezeichnete Muster zerstört. Um dieses Problem zu umgehen, wird eine Kette aus Leerstellen in der gleichen Breite wie der Bildschirm (704 Zeichenpositionen) genommen. Durch Überlagerung des Bildschirms mit diesen Leerzeichen erhalten wir sehr eindrucksvolle Effekte.

Wir können auch auf eine andere Art und Weise zu diesem Ergebnis kommen. Können Sie sich vorstellen, wie? Die naheliegendste Möglichkeit ist folgende: Wir tragen im Attribut-Bereich des Speichers direkt die jeweils gewünschte Papierfarbe ein. Der ATTR-Bereich beginnt bei 22528 und endet bei 23295. Klammert man noch die beiden unteren Zeilen aus (2 x 32 Stellen), so reicht der für die PAPER-Farbe zuständige Bereich bis 23295−64=23231. Das folgende Programm ändert jetzt die Farben des Bildschirms, ohne den Text oder das Muster zu zerstören:

```
10 FOR x = 22528 TO 23231
20 POKE x, 32
30 NEXT x
```

Erkennen Sie, welche Farbe mit POKE eingegeben wird? Da die PAPER-Farbe mit 8 multipliziert wird, wurde der Bildschirm mit der Hintergrundfarbe 4 (grün) versehen.

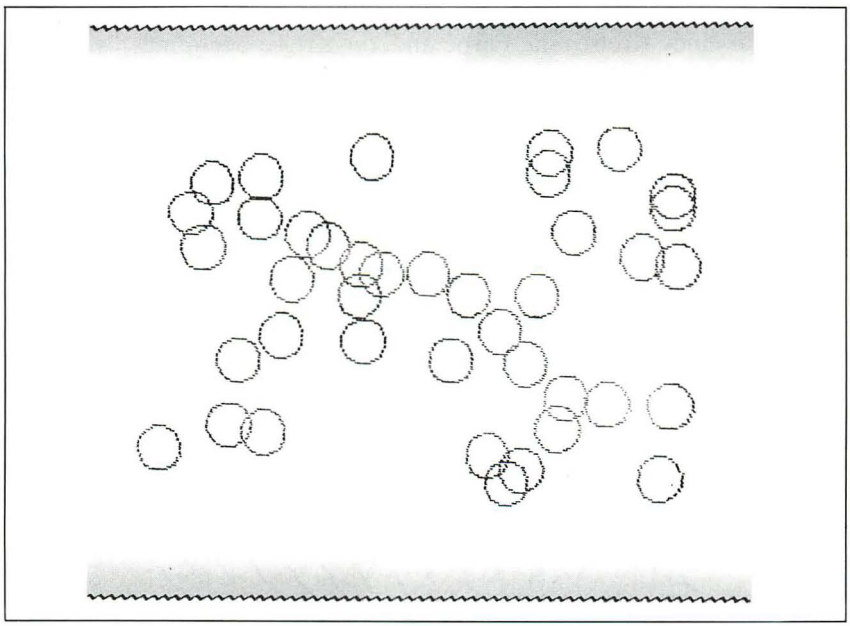
Das nächste Programm STERNENHIMMEL zeichnet ein sehr hübsches, ausgewogenes Bild auf dem Bildschirm. Dabei werden viele Farb-Befehle gegeben, die in diesem Kapitel schon besprochen wurden.

```
  10 REM STERNENHIMMEL
  20 PAPER 0: INK 7: BORDER 0: C
LS
  30 LET a=INT (127*RND)
  40 LET b=INT (87*RND)
  50 LET z=INT (8*RND)
  60 LET r=INT (6*RND)
  70 IF r<2.5 THEN GO TO 140
  80 INK z
  90 PLOT 127-a,87+b
 100 PLOT 127+a,87-b
 110 PLOT 127-a,87-b
 120 PLOT 127+a,87+b
 130 GO TO 30
 140 PLOT OVER 1;127-a,87-b
 150 PLOT OVER 1;127+a,87+b
 160 PLOT OVER 1;127-a,87+b
 170 PLOT OVER 1;127+a,87-b
 180 GO TO 30
```

Es wird Ihnen Spaß machen, farbige Kreise auf Ihrem Spectrum zu zeichnen. Der wichtigste Befehl hierbei lautet:

```
CIRCLE INK i;x,y,z
```

Mit i wird die Zeichenfarbe festgelegt, x und y sind die Koordinaten des (jeweiligen) Kreismittelpunktes, z ist der Radius. Das Programm zeichnet ZUFALLSKREISE; wie der Name schon sagt, Kreise in beliebiger Anordnung und Farbe.

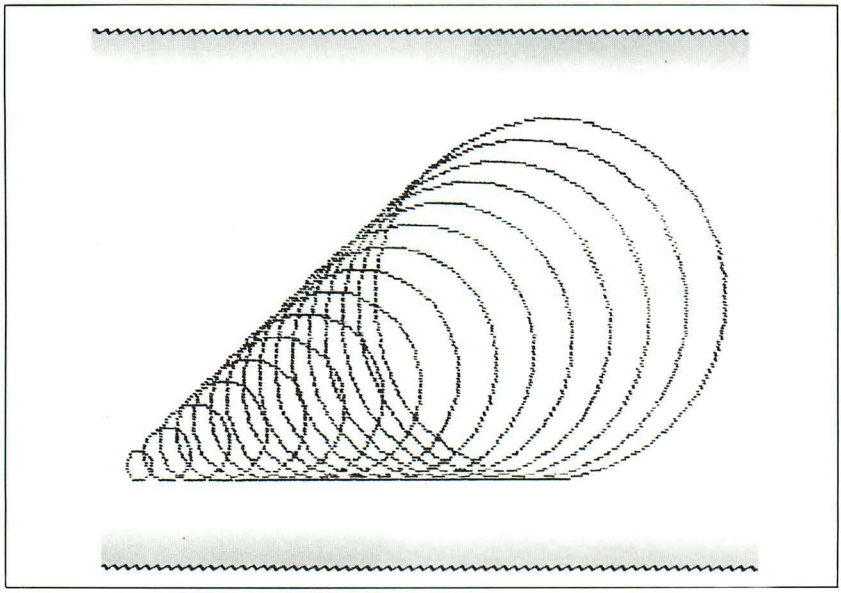


```
 1 REM ZUFALLSKREISE
 5 PAPER 0: CLS
10 LET x=(234*RND)+10
20 LET y=(154*RND)+10
30 LET z=(5*RND)+1
40 CIRCLE INK z;x,y,10
50 GO TO 10
```

Das BOWLINGKUGEL-Programm zeichnet, ausgehend von einem gemeinsamen Mittelpunkt, Kreise, deren Radius jeweils um 0.1 (STEP) größer wird. Das hat den eigenartigen Effekt, daß in den „Kugeln" fünf weiße Flecken entstehen. Daher rührt der Name dieses Programms (eine Bowlingkugel hat fünf Grifflöcher für die Wurfhand).

```
 5 INK 1: PAPER 6: CLS
10 LET x=100: LET y=x
20 FOR a=0 TO 22 STEP .1
30 CIRCLE x,y,a
40 NEXT a
50 LET x=130: LET y=85
60 GO TO 20
```

Das Zeichnen von Kreisen führt, wie wir beim Programm ZUFALLS-KREISE gesehen hatten, zu mehrfarbigen Kreisen. Das liegt daran, daß die Farbe eines Kreises die Farbe anderer Kreise beeinflußt, wenn die Kreise sehr nah aneinander liegen. Diese Wirkung macht sich das Programm KEGEL zunutze.



```
 1 REM KEGEL
10 BORDER 6
20 PAPER 0: CLS
30 LET y=30
40 FOR x=30 TO 180 STEP 10
50 LET z=(6*RND)+1
60 CIRCLE INK z;x,y,y-25
70 LET y=y+4
80 NEXT x
90 PAUSE 0
```

Wie Sie sehen, sind hier Kreise in beliebig gewählten Farben gezeichnet, einer immer größer als der andere. Dies läßt den Eindruck eines dreidimensionalen Gebildes entstehen.

Das letzte Programm könnte den Ausgangspunkt für ein Programm bilden, das noch komplexere Figuren in Farbe zeichnet. Sie können damit innerhalb der Grenzen des Bildschirms ein Dreieck beliebiger Größe zeichnen und es mit einer Farbe ihrer Wahl ausfüllen.

```
  10 REM DREIECKSZEICHNER
  20 BORDER 1: PAPER 7: CLS
  30 INPUT "START X-KOORDINATE?"
;x
  40 INPUT "START Y-KOORDINATE?"
;y
  50 INPUT "HOEHE?";h
  60 IF y+h>175 THEN GO TO 150
  70 INPUT "LAENGE?";l
  80 IF x+l>255 THEN GO TO 150
  90 INPUT "FARBE?";f
 100 CLS
 110 FOR p=0 TO h
 120 PLOT x,y: DRAW INK f;l,p
 130 NEXT p
 140 STOP
 150 CLS: PRINT "ZU GROSS! NOCH
MAL!"
 160 GO TO 30
```

Kapitel 4

Der ZX Spectrum als Musikinstrument

Sie wissen wahrscheinlich schon, wie einfach es ist, mit Ihrem Spectrum Töne zu erzeugen. Benutzen Sie den BEEP-Befehl. (Er ist rot und liegt auf der Tastatur unter der Z-Taste.) Drücken Sie zuerst die CAPS SHIFT-Taste zusammen mit der SYMBOL SHIFT-Taste auf der rechten Seite, und anschließend – während Sie die rote Shift-Taste heruntergedrückt halten – drücken Sie die Z-Taste.

Um einen Ton zu erzeugen, können Sie beispielsweise folgendes eintippen: BEEP 1,0 (danach ENTER). Sie hören dann ungefähr eine Sekunde lang einen Ton in der Tonlage des normalen C. Die erste Zahl bezieht sich auf die gewünschte Tonlänge, die zweite Zahl legt die Tonhöhe fest. Die Zahl, die die Tonlänge angibt, mißt nach Sekunden (von 0,00125 bis ca. 10). Bei einer Zeitdauer kürzer als 0,00125 könnten sie so gut wie nichts hören, und bei einer Zeitdauer von länger als 10 Sekunden würde der Computer die Fehlermeldung „ganze Zahl außerhalb der angegebenen Größenordnung" bringen.

Die zweite Zahl kann sich zwischen −60 und +69 bewegen. Jeder Schritt bezieht sich auf einen Halbtonschritt über dem normalen C (dem die Zahl 0 zugeordnet ist) oder auf einen Halbtonschritt darunter. Mit BEEP 1,1 bekommen Sie also einen Ton, der über dem normalen C liegt (= Cis). Sie hören ihn eine Sekunde lang. BEEP 1, −10 klingt 10 Halbtonschritte tiefer als das C. Vergessen Sie nicht das Komma zwischen den beiden Zahlen.

Das folgende kurze Programm gibt Ihnen eine Vorstellung von dem Frequenzspektrum Ihres ZX Spectrum:

```
10  FOR n = −60 TO 69
20  BEEP .2, n
30  NEXT n
```

Wie Sie hören, gibt es Tonlagen vom kurzen Klicken bis zum hohen Trillern. Daraus ergibt sich die Schlußfolgerung, daß am besten in den Tonlagen um das C herum Musik gemacht wird (ca. +/−20 Halbtonschritte auf jeder Seite). Das erklärt, warum Sinclair dem normalen C (großes C) den Wert 0 zuordnet. Natürlich können auch die höchste und die tiefste Tonlage sinnvoll genutzt werden, wie wir später sehen werden.

Der Ton ist sehr leise, wie Sie sicher bemerkt haben. Um die Lautstärke zu erhöhen, müssen sie den Spectrum an einen Verstärker anschließen. Die meisten Verstärker mit einer Mikro-Buchse sind geeignet. Sie hören den Ton sowohl mit dem Ohr als auch über die Mikrobuchse hinten am Spectrum. Mit dem Ohr hören Sie ihn etwas leiser. Sie müssen ein wenig herumexperimentieren, um herauszubekommen, wie Sie Ihren Verstärker am besten einsetzen.

Ist ein Verstärker angeschlossen, dann hören Sie bei jedem Druck auf eine Taste ein „Klick". Diese Rückmeldung bei jedem Tastendruck ist u. U. sehr nützlich. Leider ist dieses Klicken viel zu leise, wenn Sie keinen Verstärker angeschlossen haben. Das können Sie mit einem direkten Befehl ändern: POKE 23609, 100 (dann drücken Sie ENTER).

Jetzt hören Sie bei jedem Tastendruck einen deutlichen Piepton. Das hilft sehr, wenn Sie schnell tippen und dabei nur gelegentlich auf den Bildschirm schauen. Natürlich wird dieser Ton durch einen externen Verstärker lauter. Damit müssen Sie sich abfinden, es sei denn, Sie ziehen es vor, daß Ihr Spectrum nur leise „klickt", wenn Sie Töne benutzen. Sie können natürlich auch zwischen beiden Möglichkeiten wechseln, aber das ständige Herumhantieren mit der Verstärkerzuleitung ist ziemlich umständlich. Wenn Sie einen externen Verstärker benutzen, müssen Sie ohnehin immer die entsprechende Verstärkerzuleitung aus der Buchse nehmen, wenn Sie ein Programm speichern oder sichern wollen.

Zurück zur Musik. Jede Notenzahl beim BEEP-Befehl bedeutet einen Halbtonschritt nach oben (oder nach unten, wenn sie ein negatives Vorzeichen hat). Somit sind einer Oktave zwölf Notenzahlen zugeordnet: Das normale C ist gleich 0, das C darüber (kleines c) 12 usw. Wenn Ihr Spectrum Musik spielen soll, könnten Sie durch entsprechende Linierung des Bildschirms Ihre Melodie wie Noten auf Notenlinien eingeben. Dann ordnen Sie jede Note einer BEEP-Zahl zu und jede Tonlänge einer Zeitangabe-Zahl (0,25 Sekunden für einen Takt). Doch diese Methode ist zeitraubend. So wäre es einfacher: Sie geben die Noten der Melodie mit Buchstaben und die Takte mit Zahlen ein.

Machen Sie das bei unserem Programm MUSIKANT. Dabei haben Sie die Buchstaben A−G für die untere Oktave zur Verfügung („C" entspricht hier dem großen C) und die Buchstaben a−g für die nächst höhere Oktave. Sie geben Ihre Melodie wie folgt ein: Auf einer Zeile folgt jeder Note eine Zahl, die die Tonlänge angibt. Dabei entspricht die 1 einem

Taktschlag, die 2 zwei Taktschlägen usw. Mit folgender Zeile können Sie sich aussuchen, wie schnell Sie Ihr Musikstück gespielt haben wollen:

```
5 INPUT "Welches Tempo? 1 ist langsam, 5 ist schnell"; x
```

Sie müssen auch die Zeile mit dem BEEP-Befehl ändern:

```
370 BEEP X(Z)/X,Y(Z)
```

```
  10 REM DER MUSIKANT
  20 DIM x(50): DIM y(50)
  30 LET k=0: LET l=1
  40 BORDER 2: PAPER 4: INK 9: C
LS
  50 PRINT AT 0,10; INVERSE 1;"D
ER MUSIKANT"''
  60 PRINT "TIPPE DEINE NOTEN AL
S EINE FOLGE VON BUCHSTABEN UND
ZIFFERN EIN."
  70 PRINT '''"BENENNE DIE NOTEN
MIT IHREN BUCHSTABEN UND GIB IHR
E DAUER AN."
  80 PRINT '''"ZWEI OKTAVEN STEHE
N DIR ZUR VERFUEGUNG. DIE UNTERE
 GEHT VON A - G UND DIE OBERE VO
N a - c."
  90 INPUT n$
 100 FOR a=1 TO LEN n$ STEP 2
 110 IF CODE n$(a)<97 THEN GO TO
 280
 120 IF n$(a)="a" THEN LET k=-0.
5
 130 IF n$(a)="d" THEN LET k=0.5
 140 IF n$(a)="e" THEN LET k=1
 150 IF n$(a)="f" THEN LET k=1
 160 IF n$(a)="g" THEN LET k=1.5
 170 LET y(l)=(CODE n$(a)-87)+(2
*k)
 180 LET l=l+1
 190 LET k=0
 200 NEXT a
 210 LET l=1
 220 FOR t=2 TO LEN n$ STEP 2
 230 LET x(l)=VAL n$(t)/2
 240 LET l=l+1
 250 NEXT t
 260 GO TO 360
 270 STOP
 280 IF n$(a)="A" THEN LET k=-0.
5
 290 IF n$(a)="B" THEN LET k=-0.
5
 300 IF n$(a)="F" THEN LET k=0.5
 310 IF n$(a)="G" THEN LET k=0.5
 320 LET y(l)=(CODE n$(a)-67-k)*
2
 330 LET k=0
 340 LET l=l+1
 350 GO TO 200
 360 FOR z=1 TO LEN n$/2
 370 BEEP x(z)/2,y(z)
 380 NEXT z
```

So ist das Programm doch viel leichter durchzuführen, als wenn Sie erst einmal die Musik in Noten umgeschrieben hätten, oder?

Das Klavier

Der Traum, mit dem Spectrum Klavier spielen zu können, erfüllt sich mit Hilfe des nächsten Programms. Mit dem INKEY$-Befehl kontrolliert der Computer die Tastatur. Er „sieht", welche Taste Sie drücken. Das Programm wurde so geschrieben, daß Sie drei Oktaven zur Wahl haben. Die Note C ist dabei immer auf der T-Taste. Mit Hilfe des Bildschirms können Sie sich einen Überblick über Ihre „Klavier"-Tasten verschaffen. Auf diese Art und Weise prägen Sie sich am besten ein, mit welcher Taste welcher Ton hervorgezaubert wird. Stellen Sie sich bei den Buchstaben Q bis P die weißen Klaviertasten vor. Die Zahlen darüber bezeichnen dann die schwarzen Tasten, mit denen eine Note um einen Halbtonschritt höher oder tiefer gespielt wird. Der Spectrum ist so ausgelegt, daß er eine „wohltemperierte" Tonleiter spielen kann, die fast so wie auf einem echten Klavier klingt. Die Musik, die er produziert, klingt ganz ordentlich (wenn auch etwas leise).

```
  10 REM KLAVIER
  20 INK 7: BORDER 0: PAPER 2: C
LS
  30 PRINT AT 5,12; INVERSE 1;"K
LAVIER"
  40 PRINT AT 8,0;"DIE WEISSEN T
ASTEN DES KLAVIERS SIND DIE TAST
EN 'Q' - 'P'"
  50 PRINT AT 12,0;"WENN DU 'Z'
DRUECKST, GELANGST  DU IN EINE H
OEHERE OKTAVE. MIT  'X' GELANGST
 DU IN EINE TIEFERE"
  60 PRINT AT 16,0; INVERSE 1;"D
RUECKE 'V' FUER VIBRATO , 'M', U
M DEN LAUTSPRECHER AUSZUSCHALTEN
"
  70 PRINT AT 19,0;  INVERSE 1;"
MIT 'C' WIRD  DAS MITTLERE C
AUF DIE TASTE 'T' PLAZIERT"
  80 LET k=0: LET x=0.3
  90 PAUSE 500
 100 REM TASTATURBELEGUNG
 110 CLS
 120 PRINT ; INVERSE 1;AT 10,4;"
Q";AT 10,6;"W";AT 10,8;"E";AT 10
,10;"R";AT 10,12;"T";AT 10,14;"Y
";AT 10,16;"U";AT 10,18;"I";AT 1
0,20;"O";AT 10,22;"P"
 130 PRINT PAPER 0;AT 8,5;"2";AT
 8,7;"3";AT 8,9;"4";AT 8,13;"6",
AT 8,15;"7";AT 8,19;"9";AT 8,21,
"0"
 140 IF INKEY$="Z" THEN LET k=12
 150 IF INKEY$="X" THEN LET k=-1
2
```

```
 160 IF INKEY$="C" THEN LET k=0
 170 IF INKEY$="V" THEN LET k=0.
00
 180 IF INKEY$="M" THEN LET k=0.
5
 190 IF INKEY$="2" THEN BEEP x,-
6+k
 200 IF INKEY$="3" THEN BEEP x,-
4+k
 210 IF INKEY$="4" THEN BEEP x,-
2+k
 220 IF INKEY$="6" THEN BEEP x,1
+k
 230 IF INKEY$="7" THEN BEEP x,3
+k
 240 IF INKEY$="9" THEN BEEP x,6
+k
 250 IF INKEY$="0" THEN BEEP x,8
+k
 260 IF INKEY$="Q" THEN BEEP x,-
7+k
 270 IF INKEY$="W" THEN BEEP x,-
5+k
 280 IF INKEY$="E" THEN BEEP x,-
3+k
 290 IF INKEY$="R" THEN BEEP x,-
1+k
 300 IF INKEY$="T" THEN BEEP x,0
+k
 310 IF INKEY$="Y" THEN BEEP x,2
+k
 320 IF INKEY$="U" THEN BEEP x,4
+k
 330 IF INKEY$="I" THEN BEEP x,5
+k
 340 IF INKEY$="O" THEN BEEP x,7
+k
 350 IF INKEY$="P" THEN BEEP x,9
+k
 360 GO TO  130
 370 STOP
```

Das Programm fragt Sie, welche Oktave Sie wünschen, wobei Sie, wie gesagt, beim Programmlauf zwischen drei Oktaven wählen können. Das jeweilige C liegt in jedem Fall auf der T-Taste. Natürlich können Sie auch die Zeilen 140 bis 350 neu schreiben. Dann bekommen Sie mit anderen Tasten andere Töne. Als letztes werden Sie gefragt, ob Sie ein Vibrato wollen. In diesem Fall wird die Tonlänge von 0.3 auf 0.03 festgelegt. Versuchen Sie es auch einmal mit anderen Tonlängen, um die Wirkung zu sehen. Meines Erachtens wirkt das Vibrato bei Tonlängen von sehr viel mehr als ca. einer Sekunde zu langsam, während es bei Tonlängen von weniger als 3/100 Sekunden zu sehr „klickt". Vielleicht möchten Sie einmal das Tempo ändern und Varianten wie „langsam, schnell, schnell" ausprobieren. Dann müssen Sie die Tonlänge jeder Note ändern. Versuchen Sie ein Unterprogramm mit dem GO SUB-Befehl. Dazu müßten Sie erst die gewünschte Note in der Ausgangstonlänge spielen und dann zwei

Noten, die nur 1/2 oder 1/4 so lang sind wie die erste. Denken Sie aber bitte daran, daß Unterprogramme Zeit brauchen. Wenn Sie das ganze Programm mit zuviel Sonderwünschen aufblähen, reagiert es etwas schwerfällig.

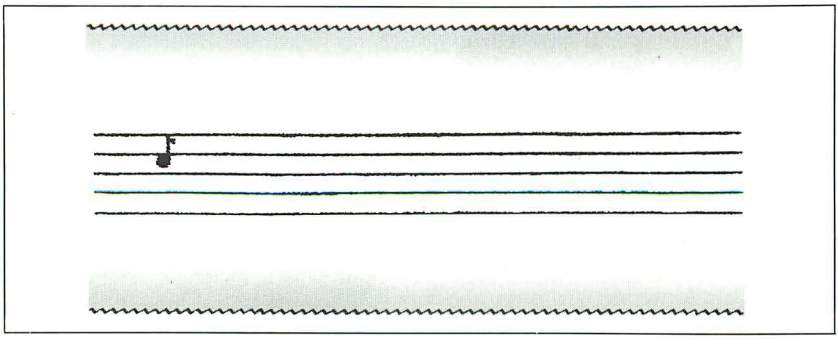
Der Spectrum spielt nicht nur kürzeste Tonlängen, sondern auch kleinste Tonschritte. Machen Sie sich das zunutze, wenn Sie Ihren Spectrum auf ein anderes Instrument einstimmen wollen. Das läßt sich ganz einfach bewerkstelligen, indem Sie das Programm um folgende Zeilen erweitern:

```
355  IF INKEY$ = "K" THEN LET K=K+0.05
356  IF INKEY$ = "L" THEN LET K=K-0.05
```

Mit jedem Druck auf die K- oder L-Taste (K für höher, L für tiefer) wird die betreffende Note einen Bruchteil höher odere tiefer gespielt. Mit dieser Methode ist es sogar möglich, durch minimale Tonschwankungen einen „orientalischen" Klang zu erzeugen. Die Tonleiter umfaßt dann nicht zwölf, sondern 16 Noten und mehr.

Andere Klangeffekte

Sie können mehr mit dem Spectrum machen, als ihn „nur" in ein Klavier zu verwandeln. Sie können die Noten sichtbar machen. Dazu brauchen wir die Fähigkeit des Spectrum, neue Zeichen zu definieren (die eingehende Erläuterung finden Sie im Kapitel „Der Spectrum als Spielpartner"). Das Programm KOMPONIST bildet die Grundlage für ein längeres Programm, bei dem Sie Ihre Musik wie im Programm MUSIKANT in Form von Buchstaben, Zahlen und Zeichen eingeben können. Aber jetzt sehen Sie jede Note als Viertel- oder Achtelnote u. a. auf einer richtigen Notenlinie. Sie können das Programm so ändern, daß Sie, wenn Sie Orgel oder Klavier spielen, die Noten auf dem Bildschirm sehen.



```
  10 REM KOMPONIST
  20 PAPER 6: INK 0: BORDER 7: C
LS
  30 LET x=151
  40 REM ZEICHNEN NOTENLINIEN
  50 FOR y=1 TO 5
  60 PLOT 0,x: DRAW 255,0
  70 LET x=x-8
  80 NEXT y
  90 FOR a=0 TO 7
 100 READ t
 110 POKE USR "L"+a,t
 120 NEXT a
 130 FOR b=0 TO 7
 140 READ t
 150 POKE USR "K"+b,t
 160 NEXT b
 170 PRINT OVER 1;AT 3,3;"L";AT
4,3;"K"
 180 BEEP 0.25,0
 190 REM NOTENHALS
 200 DATA 0,4,7,5,4,4,4,4
 210 REM NOTE
 220 DATA 0,60,124,124,124,56,0,
0
 230 STOP
```

Zusätzliche Klangeffekte

Höchstwahrscheinlich haben Sie Spaß an Computerspielen. Sie fänden es sicher faszinierend, wenn Sie noch zusätzlich Geräusche von Feuersalven, Fußschritten oder Zügen einbauen können. Oder etwa nicht? Solche und ähnliche Geräusche könnten den Reiz einiger Spiele erheblich steigern. Andererseits müssen Sie bedenken, daß ein Computer nicht so wirklichkeitsgetreu Geräusche produzieren kann wie ein Musikautomat.

Mit den Programmen FEUER 1, FEUER 2 und FEUER 3 können Sie ganz gut Geräusche für Spiele herstellen, bei denen eine Schießerei im Weltall stattfindet.

```
10 REM FEUER 1
20 LET d=0.0125
30 FOR x=1 TO 2
40 FOR y=4 TO 16 STEP 2
50 BEEP d,y
60 NEXT y
70 NEXT x

10 REM FEUER 2
20 FOR x=-10 TO 0
30 BEEP 0.0125,x
40 NEXT x
50 FOR y=0 TO -5 STEP -1
60 BEEP 0.0125,y
70 NEXT y
```

```
10 REM FEUER 3
20 FOR x=5 TO 20 STEP 1.5
30 BEEP .008,x
40 NEXT x
50 FOR y=20 TO 5 STEP -1.5
60 BEEP .008,y
70 NEXT y
80 PAUSE 30
90 RUN
```

Das Programm SPAZIERGANG veranschaulicht, wie Sie Geräusch und Bewegung miteinander verbinden können. Beachten Sie bitte, daß in Zeile 140 das erste grafische Zeichen ein L, das zweite ein K ist.

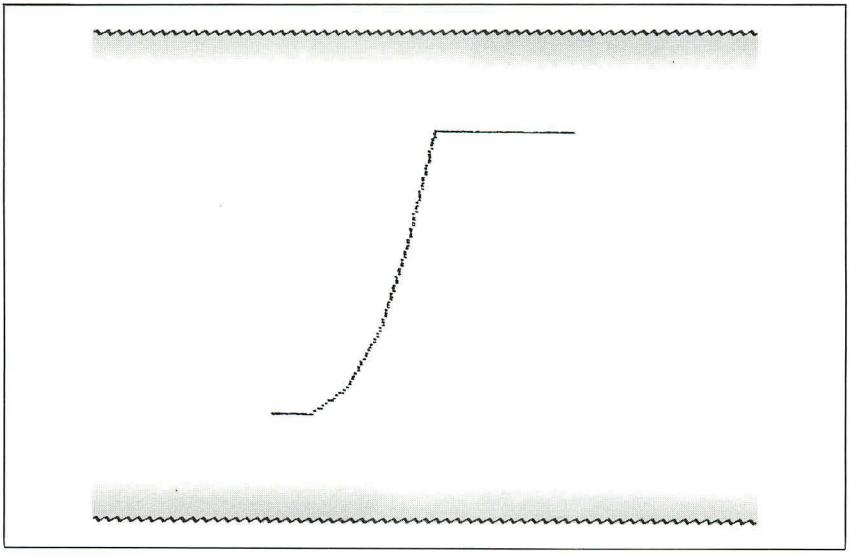
```
10 REM SPAZIERGANG
20 FOR a=0 TO 7
30 READ x
40 POKE USR "L"+a,x
50 NEXT a
60 FOR a=0 TO 7
70 READ x
80 POKE USR "K"+a,x
90 NEXT a
100 FOR t=0 TO 31
110 PRINT AT 21,t;"^";
120 NEXT t
130 FOR c=0 TO 31
140 PRINT AT 20,c;"L": PAUSE 3:
 PRINT AT 20,c;"K": PAUSE 3: PRI
NT AT 20,c;" "
150 BEEP 0.02,30: BEEP 0.02,40
160 NEXT c
170 GO TO 100
180 REM DATA FUER 2 FIGUREN
190 DATA 24,36,153,126,24,100,1
02,4,24,36,153,126,24,38,33,32
200 STOP
```

Im nächsten Programm, das den Abwurf einer Bombe simuliert, wird das Geräusch des Fallens dadurch erzeugt, daß die Noten(Ton-)frequenz vom höchstmöglichen Ton sehr schnell um 20 Halbtöne fällt. Die damit simulierte Fallgeschwindigkeit ist von Anfang an sehr groß, nimmt aber noch während des Falles an Geschwindigkeit zu. Sie hören förmlich, wie die Bombe schneller wird, je näher sie dem Boden kommt.

```
10 REM BOMBE
20 FOR x=69 TO 55 STEP -0.3
30 BEEP 0.05,x
40 NEXT x
50 FOR y=0 TO 20
60 BEEP 0.01,-10: BEEP 0.01,-5
0: BEEP 0.01,-60
70 NEXT y
```

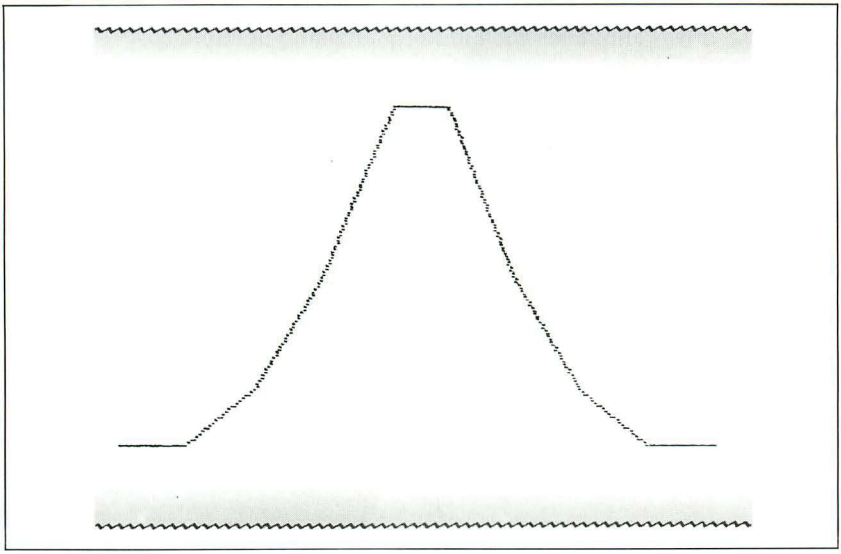
Mikrocomputer, die noch weiter entwickelt sind, können nicht nur einzelne Noten, sondern dreistimmige Musik spielen. Sie variieren dabei die Frequenz (innerhalb eines Frequenzbandes). Manche Computer haben auch einen Rausch-Generator für brandungsähnliche Geräusche. Der Spectrum hat diese Möglichkeiten nicht. Aber mit ein paar Tricks produziert er immerhin mehr als einfach nur BEEP, BEEP-Laute. Mit den Programmen FREQUENZ/ZEIT- und FREQUENZ-DIAGRAMME können Sie so etwas ähnliches wie ein Frequenzband herstellen.

Im ersten Programm können Sie die Schrittweite zwischen zwei Noten sowie die Dauer der Noten variieren. Im vorliegenden Beispiel geht das mittlere C zur nächsthöheren ganzen Note. Es beginnt mit einer relativ kurzen Tondauer und wird immer kürzer, je mehr es sich dem oberen Ton nähert.



```
  10 REM FREQUENZ/ZEIT-DIAGRAMME
  20 INK 7: BORDER 6: PAPER 1: C
LS
  30 LET dauer=0.05
  40 FOR f=0 TO 2 STEP 0.5
  50 BEEP dauer,f
  60 DRAW 300*dauer,20*f
  70 LET dauer=dauer-0.005
  80 NEXT f
  90 DRAW 50,0: BEEP 0.5,2
 100 GO TO 100
```

Das Diagramm des zweiten Programms steigt in Schritten, die Sie in Zeile 40 genau festlegen, von einer Frequenz zur nächsten rampenartig an. So springt der Klang nicht unvermittelt von einer Tonhöhe zur nächsten. Er steigt und fällt vielmehr langsam, weil die Tonhöhe immer nur minimal verändert wird.



```
  10 REM FREQUENZDIAGRAMM
  20 BORDER 4: PAPER 4: CLS
  30 REM DIE VARIABLEN f,fe UND
s DUERFEN GEAENDERT WERDEN
  40 LET f=0: LET fe=3: LET s=1
  50 FOR x=f TO fe STEP s
  60 DRAW 25*s,20*x
  70 BEEP 0.03,x
  80 NEXT x
  90 DRAW 20,0: BEEP 0.2,fe
 100 FOR y=fe TO f STEP -s
 110 DRAW 25*s,-20*y
 120 BEEP 0.03,y
 130 NEXT y
 140 GO TO 140
```

Vielleicht möchten Sie mit diesen Frequenzbändern ein wenig herumexperimentieren und beispielsweise mehrere miteinander verbinden, um komplexe Tonfolgen aufzustellen. Dadurch würde der Klang des Spectrum noch leiser, so daß ich Ihnen den Einsatz eines externen Verstärkers empfehle.

Das Schlußprogramm MUSIKALISCHE MALEREI verbindet die Fähigkeiten des Spectrum, Klang und Farbe zu erzeugen. Es läßt Noten von beliebiger Tonhöhe und -länge erklingen (wobei Grenzwerte angegeben sind). Je höher die Note, desto weiter oben entsteht auf dem Bildschirm ein Farbklecks. Und je länger die einzelne Note, desto mehr nach rechts plaziert der Spectrum seinen Klecks. Dadurch entsteht eine Farbcollage, bei der das, was Sie sehen, in direkter Beziehung zu dem steht, was Sie hören.

```
 10 REM MUSIKALISCHE MALEREI
 20 BORDER 0: PAPER 0: CLS
 30 FOR x=0 TO 31
 40 LET note=RND*20
 50 LET dauer=RND*0.4
 60 LET tinte=(RND*6)+1
 70 PRINT INK tinte;AT 20-note,
x;"■"
 80 BEEP dauer,note
 90 NEXT x
100 CLS : GO TO 30
```

Kapitel 5

Der Gebrauch des Spectrum im kaufmännischen Bereich

Wie alle Computer kann der Spectrum nicht selbstständig denken und handeln. Dagegen kann er mit entsprechender Software ein breites Spektrum von Arbeiten erledigen. Benutzer kaufen ihre Software mit der Vorstellung, daß vieles davon eigentlich überflüssig oder für ihre Zwecke unnötig ist. Aber gerade diese Teile der Software sind unabdingbar notwendig. Sie merken das erst später (wenn das System gut ist) und verstehen dann nicht mehr, wie Sie jemals ohne diese Softwarezusätze auskommen konnten. Einige Beispiele für das, was ein Computersystem leisten kann, sind im folgenden kurz erläutert. Sie werden schnell sehen, daß Computersysteme, wenn sie gut sind, eine enorme Hilfe sein können.

Wie schon im ersten Kapitel gesagt wurde, können Computer alles ausführen, was Sie ihnen durch Ihre Befehle angeben. Sie richten sich immer genau nach Ihren Anweisungen. Wenn Ihr Programm, das Rechnungen bearbeitet, z. B. den Befehl erhält: „Schicken Sie jedem eine Mahnung, der in diesem Monat seine Rechnung nicht bezahlt hat", kann es vorkommen, daß auch Mahnungen über DM 0,00 ausgestellt werden, sogar mit der Androhung rechtlicher Schritte bei Nichtbezahlung.

Wie sorgfältig Sie bei der Herstellung und Benutzung von Computerprogrammen vorgehen müssen, hängt davon ab, zu welchem Zweck Sie es einsetzen wollen. Die Anstrengung, die es kostet, wenn Sie beim Programmieren hundertprozentig sicherstellen wollen, daß die Raumfähre z. B. nach oben und nicht nach unten fliegt, wäre für ein Textbearbeitungsprogramm völlig überflüssig. Der Aufwand bei der Programmierung sollte also immer dem Zweck angepaßt sein.

Die Anwendungsmöglichkeiten des Computers kann man in Gruppen zusammenfassen (wobei die Grenzen fließend sind):

Datenverarbeitung im kaufmännischen Bereich

Normalerweise werden recht einfache Arbeitsgänge mit einer ziemlich großen Datenmenge durchgeführt. Für eine Kontrolle der Aktienkurse

werden im Grunde nur Addition und Subtraktion gebraucht. Allerdings müssen die dazu notwendigen Daten so schnell wie möglich zur Hand sein. Das ist der springende Punkt.

Beim Durchlauf eines Programms ist die Verarbeitung gespeicherter Informationen besonders zeitraubend. Ein gut ausgearbeitetes „Geschäftsplanungsprogramm" besteht aus Modulen, die wechselseitig aufeinander einwirken. Jeder Verkauf müßte sowohl im Hauptbuch als auch beim Warenausgang zu Änderungen führen und Rechnungen, Bestellungen u. ä. zur Folge haben. Es ist von Vorteil, wenn Ihr Textverarbeitungsprogramm Ihr Geschäftsplanungsprogramm beeinflußt und umgekehrt. Alles sollte so narrensicher wie möglich sein. Vor allem sollte Irrtum erwartet und möglichst schon im Frühstadium berücksichtigt werden (z. B. sollte das Programm schon bei der Erfassung prüfen, ob die Daten einigermaßen vernünftig sind). Von solchen „Geschäftsplanungsprogrammen" gibt es nur wenige, aber die wenigen sind ihr Geld wert.

Gute Software ist teuer in der Herstellung und in der Wartung. Z. B. kann ein Spectrum-Paket, das zur Zeit gekauft werden kann, ziemlich anspruchsvolle Projekte planen. Es ist so gut wie nur irgendein vergleichbares Programm, das zur Zeit angeboten wird, und kostet weniger als andere Planungsprogramme. Allerdings kostet es mehr als ein 16K Spectrum. Wenn Sie es genau betrachten, ist der Aufwand bei der Auswahl eines geeigneten Programmpaketes höher als beim eigentlichen Kauf. Es wird viel Zeit in Anspruch nehmen, einem Anwender zu erklären, daß der geforderte Preis gerechtfertigt ist. Nur bei angemessenem Preis kann wirklich erstklassige Software auf den Markt kommen.

Textverarbeitung

Mit einer zusätzlichen Tastatur kann der Spectrum recht gut zur Textverarbeitung verwendet werden. Sie sehen Ihren getippten Text auf dem Bildschirm. Sie können ihn ändern, etwas einfügen (INSERT) oder auslöschen (DELETE). Sie können sogar den Text vollständig neu ordnen – ganz wie Sie wollen. Wenn Sie mit dem Text zufrieden sind, drucken (PRINT) Sie ihn aus. An die äußere Form hat der Computer schon gedacht – der Text sieht sauber und ordentlich aus. Sie können ihn jetzt noch einmal durchgehen und Änderungen anbringen.

Wissenschaftliche Berechnungen

Derartige Berechnungen können sehr komplex sein. Dazu werden relativ wenige Eingaben, Ausgaben und Speicherplatz benötigt. Doch wird für den eigentlichen Rechenvorgang viel Zeit verwendet.

Technische Anwendungen

Der Spectrum kann das Leben des Architekten, Planers und Ingenieurs erleichtern. Grundkenntnisse beim Programmieren zahlen sich gerade

bei diesen Anwendungen aus, da es auf diesem Gebiet Berechnungen gibt, die nur einmal vorkommen und nicht dem Standard entsprechen. Es gibt natürlich auch Standard-Software für technische Probleme, und sie ist oft ihren Preis wert.

Echtzeit- und Kontrollprogramme

Bei diesen Programmen wird der Computer in die „Realität gestoßen": Am Ende eines Durchlaufs wird kein Ausdruck auf dem Bildschirm, sondern eine Handlung erwartet, wie z. B. Motoren an- oder ausstellen. Oft sind diese Programme zeitkritisch, d. h. es muß etwas zu einer festgelegten Zeit stattfinden, weder zu früh noch zu spät. Darin unterscheiden sich diese Programme von anderen Programmen, die nach ihren eigenen zeitlichen Abläufen arbeiten.

Kommunikationen

Computer können mit einer Vielzahl von Geräten zusammenarbeiten, sogar mit Computern. Dazu werden nur normale Telefonleitungen benötigt, und schon eröffnen sich viele interessante Möglichkeiten.

Der Spectrum ist sehr gut für geschäftliche Belange geeignet. Sie bekommen mit ihm eine leistungsfähige Maschine zu einem vernünftigen Preis. Wenn in einem Geschäft ein Computer erforderlich ist, dann sprechen viele handfeste Gründe für seine Anschaffung. Fast jeder, der eine Schreibmaschine benutzt, braucht eine Textverarbeitungsmaschine. Der Spectrum kann auch bei Echtzeit-Anwendungen und bei wissenschaftlicher Arbeit eingesetzt werden.

Im allgemeinen ist es empfehlenswert, sich die Software, die Sie benötigen, anzusehen, bevor Sie sich einen Computer anschaffen. Viele Personen waren fasziniert und kauften sich einen Computer mit einer eindrucksvollen Bandbreite von Anwendungsmöglichkeiten. Sie mußten schließlich entdecken, daß sie keine Software dafür kaufen konnten und daß es ziemlich teuer gekommen wäre, sie extra in Auftrag zu geben. Dieses Mißtrauen ist beim Spectrum auf Grund seines hohen Marktanteils nicht angebracht. Was immer Sie für ihn brauchen, können Sie jetzt bekommen bzw. wird in Kürze erhältlich sein. Hinzu kommt, daß Spectrum-Software generell preiswerter angeboten wird als Software für andere Maschinen. Ein Grund dafür ist, daß bei ihm langatmige Erläuterungen, Kundenunterstützung oder individuelle Anpassungen selten vorkommen. Denken Sie daran, wenn Sie Software für den Spectrum aussuchen, daß die Qualität von Verpackung und Werbung nicht unbedingt mit der Qualität der Software übereinstimmen muß. Vor allem sollten Sie Ihre Informationen aus Empfehlungen erfahrener Benutzer und, bis zu einem gewissen Grad, auch aus Zeitschriften beziehen. Vielleicht wollen Sie im Augenblick noch keine Software für kommerzielle Anwendungen kaufen. Trotzdem können Sie sich von Ihrem Computer jetzt schon zeigen

lassen, wie viele Anwendungsmöglichkeiten es für die Programme gibt, die Sie selbst geschrieben oder überarbeitet haben.

In diesem Kapitel soll Ihnen mit fünf Programmen gezeigt werden, wie Ihnen der Spectrum zu Hause oder in Ihrem Geschäft helfen kann. Möglicherweise entsprechen diese Programme nicht völlig Ihren Vorstellungen. Keine Sorge, sie lassen sich leicht abändern und Ihren Bedürfnissen anpassen.

Folgende Programme wurden entwickelt:

Privatausgaben – damit können Sie Ihre laufenden Ausgaben genau aufschlüsseln. Wie andere Programme enthält dieses Programm ein Menü, das Ihnen u. a. erlaubt, den gegenwärtigen Stand Ihrer Konten auf Kassette abzuspeichern.

Terminkalender – dieses Programm sortiert die Eingaben nach Datum und druckt sie auf Anfrage aus.

Telefonverzeichnis – es nimmt Namen und Telefonnummern auf und sucht im Bedarfsfall den gewünschten Namen aus dem Verzeichnis heraus. Es kann jederzeit auf den aktuellen Stand gebracht und ergänzt werden.

Datenbank – sie speichert bis zu 130 Angaben mit je 24 Zeichen zweimal (dies gilt für die 16K-Version; auf einem 48K-Spectrum kann eine größere Datei angelegt werden). Auf diese Art und Weise kann die Datei nach einem von zwei Kriterien durchsucht werden. Mit diesem Programm könnten Sie z. B. eine Sammlung von 130 Schallplatten speichern. In alphabetischer Reihenfolge der Künstlernamen oder (Platten-)Titel kann eine Liste aufgestellt und auf den Bildschirm geschrieben werden, nach der Sie dann das Gewünschte heraussuchen können. Das Programm läßt sich nötigenfalls leicht ändern, wenn Sie z. B. nur eine einzige Datei oder weniger Zeichen pro Datei brauchen. Dann könnte das Programm mit viel mehr Angaben und auch günstiger arbeiten (die Reihenfolge der Suchverfahren bleibt dabei unverändert).

Finanzplan/Verkaufsvorhersage – Ein weltweit bekanntes Softwarepaket für die Geschäftsplanung ist VisiCalc, ein tabellarisches Kalkulationsprogramm. Der Benutzer kann aktuelle Aufgaben eingeben und davon ausgehend Pläne für die Zukunft entwickeln. Das folgende Programm stellt eine der Aufgaben von VisiCalc vor: Es analysiert den Anfall der Verkaufszahlen innerhalb eines Monats (oder welchen Zeitraum auch immer Sie haben wollen). Bei Planungen mit diesen Angaben können Sie dann entweder vom durchschnittlichen Absatz pro Monat für diesen Zeitraum ausgehen, sofern die Ergebnisse bekannt sind, oder von den Verkaufsergebnissen des letzten bekannten Monats.

Diese Programme müssen vielleicht noch etwas abgeändert werden, bevor sie für Sie zu Hause oder in einem kleinen Geschäft von Nutzen sind.

Davon abgesehen sollen sie vor allem zeigen, wie Geschäftsplanungssoftware geschrieben werden kann. Im Gegensatz zu Spielautomatenprogrammen, die sich durch besonders trickreiche Programmierung auszeichnen, werden Geschäftsprogramme übersichtlich und gut lesbar geschrieben. Das liegt daran, daß sich die Aufgabenstellung eines Geschäftsprogramms schon vorher recht klar umreißen läßt, während ein Spielprogramm erst im Lauf der Programmierung entwickelt wird.

Wenn Sie wissen, was Sie mit einem Geschäftsprogramm machen wollen, werden Sie sehr schnell ein Programm schreiben können oder jemanden dafür finden. Jetzt schauen wir uns die Programme nacheinander an. Bei genauer Betrachtung aller Listen können Sie vielleicht lernen, wie Sie Geschäftsanwendungen programmieren können.

Private Ausgaben

Beim Start dieses Programms sehen Sie zuerst folgendes Menü auf dem Bildschirm:

Gegenwärtiger Saldo ist 0 DM

Eingabe:
1 – Start von Anfang an
2 – Zahlungs-Tabelle ändern
3 – Geld einzahlen
4 – Auf Band speichern
5 – Programmende

Mit solch einem Menü gehen Sie immer sicher, daß das Programm ohne großartige Instruktionen durchgeführt werden kann und ohne einen Programmierer, der Ihnen erklärt, was Sie als nächstes tun sollen. Vernünftigerweise geben Sie beim ersten Programmstart die 1 ein: „Von Anfang an durchlaufen lassen". Dann erscheint auf dem Bildschirm: „Geben Sie die Anzahl der Posten ein, die jeden Monat bezahlt werden müssen." Jetzt geben Sie an, aus wie vielen Posten sich Ihre Fixkosten zusammensetzen, wie z. B. Ihre Hypotheken, die Bezahlung Ihres Autos, irgendwelche Ratenzahlungen, Ihre Kreditkarte, andere Daueraufträge u. ä. Geben Sie beim ersten Durchlauf lieber zuviel Posten an. Nach dem Programmstart können Sie nämlich nichts mehr hinzufügen.

Wenn Sie die 4 eingeben, werden Sie aufgefordert, der Reihe nach die Bezeichnung für jeden einzelnen Posten der Fixkostenaufstellung anzugeben, zusammen mit dem jeweils monatlich fälligen Betrag (genauso zeigt es das Beispiel, das der Programmliste folgt). Diese Angaben können später leicht geändert werden, wenn Sie erst einmal die Fixkostenaufstellung vollständig durchgegangen sind (dabei setzten Sie dort Blankoangaben ein, wo Sie über Ihre gegenwärtigen Bedürfnisse hinaus noch nicht vorhandene Fixkostenpositionen mitgezählt haben). Am Ende der Fixkostenaufstellung erhalten Sie nämlich vom Computer die Rückmeldung:

„Wenn alles stimmt, drücken Sie ENTER. Wenn nicht, geben Sie die Nummer der einzelnen Angabe an, die Sie ändern wollen."

Falls das Programm Ihren Banksaldo nicht kennt, z.B. beim ersten Durchlauf, meldet es „Geben Sie den letzten bekannten Saldo ein" und dann: „Geben Sie nacheinander alle Einzahlungen ein, wie z.B. Ihr Gehalt. Am Ende der Eingaben schreiben Sie ‚E'." Bei Einzahlungen erhöht sich automatisch die Saldosumme (links oben auf dem Bildschirm zu sehen). Wenn Sie die Eingabe aller Einzahlungen mit „E" signalisiert haben, müssen Sie sich wieder einmal zwischen zwei Möglichkeiten entscheiden: „Geben Sie mit 1 den Befehl zur Subtraktion der gegenwärtigen Kostenaufstellung oder tippen Sie 2 für das Menü." Bei der Eingabe von 1 subtrahiert der Computer den gesamten Endbetrag aller vorhin aufgelisteten Fixkostenposten. Mit dem Ausdruck des Saldos sind alle monatlich anfallenden Kosten subtrahiert worden.

Sie sind jetzt wieder beim Start-Menü. Jetzt ist der „gegenwärtige Saldo..." (hoffentlich) nicht mehr gleich Null wie beim ersten Durchlauf. Wenn Sie das Programm auf Kassette speichern, werden alle Variablen mitgespeichert. Wenn Sie allerdings beim nächsten Mal das Programm mit RUN oder CLEAR starten, wird der Inhalt gelöscht. Die letzte Zeile des Programms (GO TO 430) stellt sicher, daß es nach dem ersten Programmdurchlauf automatisch startet, ohne den augenblicklichen Saldostand zu löschen. Beachten Sie bitte beim Programmdurchlauf, wie Sie INK und FLASH benutzen, um Teile des Programms hervorzuheben. Farbe und andere grafische Effekte sollten dazu dienen, jeden Fehler durch den Benutzer nach Möglichkeit auszuschalten. Ebenso sollte schon der Input möglichst viele „Fehlerfallen" enthalten, damit kein Benutzerfehler das ganze Programm zusammenbrechen läßt. Z.B. akzeptiert die Zeile 325 keine Null-Eingabe bei Zeile 320. Eine Leer-Eingabe (dazu drücken Sie ENTER, ohne vorher eine Zahl anzugeben) würde in Zeile

```
   10 REM PRIVATAUSGABEN
   12 LET saldo=0
   15 GO TO 430
   20 PRINT '''"WIEVIEL POSTEN FA
LLEN MONATLICH AN?"
   30 INPUT zahl: CLS
   40 DIM a$(zahl,12)
   50 DIM a(zahl)
   60 FOR a=1 TO zahl
   70 PRINT AT a-1,0;"NAME DER KO
STENART "; FLASH 1;a
   80 INPUT a$(a)
   90 PRINT AT a-1,0;"WIE HOCH SI
ND DIE KOSTEN PRO MONAT IN DM?"
  100 INPUT a(a)
  110 PRINT AT a-1,0;a;TAB 3;a$(a
);TAB 15;"DM";a(a);"
```

```
 120 NEXT a
 130 PRINT '"TIPPEN SIE ", FLASH
 1;"ENTER"; FLASH 0;", WENN DIES
 RICHTIG IST, ODER GEBEN SIE DI
E NR. DER FEHLERHAFTEN KOSTENART
 AN!"
 140 INPUT b$
 150 IF b$="" THEN GO TO 260
 160 LET b=VAL b$
 170 PRINT "NEUE BEZEICHNUNG FUE
R DIE KOSTENART ";b
 180 INPUT a$(b)
 190 PRINT "UND WIE HOCH SIND DI
E KOSTEN PRO MONAT IN DM?"
 200 INPUT a(b)
 210 CLS
 220 FOR a=1 TO zahl
 230 PRINT a;TAB 3;a$(a);TAB 15;
"DM";a(a)
 240 NEXT a
 250 GO TO 130
 260 CLS
 265 IF saldo<>0 THEN GO TO 290
 270 PRINT '"WIE WAR DER LETZTE
 KONTOSTAND?"
 280 INPUT saldo
 290 CLS
 300 PRINT AT 0,0;"DM";saldo
 310 PRINT '"GELDEINGANG IM EIN
ZELNEN, Z.B. GEHALT ETC.","ENDE
BEI EINGABE VON "; FLASH 1;"E"
 320 INPUT q$
 325 IF q$="" THEN GO TO 320
 330 IF q$="E" OR q$="e" THEN GO
 TO 360
 340 LET saldo=saldo+VAL q$
 350 GO TO 300
 360 INPUT "TIPPEN SIE 1, UM ALL
ES VOM LAUFENDEN KOSTENPLAN ZU S
UBTRAHIEREN, ODER 2, UM INS MENU
E ZU GEHEN!";c
 365 IF c=2 THEN GO TO 450
 370 LET ausgaben=0: FOR a=1 TO
zahl
 380 LET ausgaben=ausgaben+a(a)
 390 NEXT a
 400 CLS
 410 PRINT '"SALDO VOR LAUFENDE
M KOSTENABZUG WAR DM ";saldo
 420 LET saldo=saldo-ausgaben
 430 PRINT '"AKTUELLER STAND IS
T DM ";saldo: PAUSE 200
 450 PRINT '"AUSWAHL:"
 460 PRINT '"1 - NOCHMAL NEU ANF
ANGEN"
 470 PRINT '"2 - KOSTENPLAN AEND
ERN"
 480 PRINT '"3 - GELDBEWEGUNGEN"
 485 PRINT '"4 - SPEICHERN AUF K
ASSETTE"
 490 PRINT '"5 - STOP"
 500 INPUT c: CLS
 510 IF c=1 THEN LET saldo=0: GO
 TO 20
 520 IF c=2 THEN GO TO 210
 530 IF c=3 THEN GO TO 300
 540 IF c=4 THEN SAVE "AUSGABEN"
 550 IF c=5 THEN STOP
 560 GO TO 430
```

340 zu einem Fehler führen. Genauso akzeptiert das Programm den Kleinbuchstaben „e", um die Eingabe der Einzahlungen abzuschließen. Das Programm will zwar (in Zeile 310) das große „E", aber innerhalb des Programms gibt es keinen Befehl, der sicherstellt, daß CAPS LOOK festgelegt wurde.

Probelauf: PRIVATE AUSGABEN

```
AKTUELLER SALDO IST DM1342.75

AUSWAHL:

1 - NOCHMAL NEU ANFANGEN

2 - KOSTENPLAN AENDERN

3 - GELDBEWEGUNGEN

4 - SPEICHERN AUF KASSETTE

5 - STOP

1  MIETE     DM350
2  HYPOTHEK  DM500
3  AUTO      DM42.75
4  SONSTIGE  DM100

TIPPEN SIE ENTER, WENN DIES RICH
TIG IST, ODER GEBEN SIE DIE NR.
DER FEHLERHAFTEN KOSTENART AN!

DM1342.75

GELDEINGANG IM EINZELNEN, Z.B.
GEHALT ETC.
ENDE BEI EINGABE VON E

SALDO VOR LAUFENDEM KOSTENABZUG
WAR DM 1342.75

AKTUELLER SALDO IST DM 400

AUSWAHL:

1 - NOCHMAL NEU ANFANGEN
```

Terminkalender

Das Terminkalenderprogramm kann natürlich für jedes Planungsproblem benutzt werden. Obwohl es im Augenblick mit Tagen arbeitet, können Sie es ziemlich leicht abändern, um gegebenenfalls mit Stunden zu arbeiten.

Dieses Programm ist das einfachste in diesem Kapitel. Seine Aufgabe ist: (a) ein Datum und ein Ereignis zur Kenntnis zu nehmen; (b) die Ereignisse in der Reihenfolge ihres Datums zu ordnen und (c) sie auf eine Anfrage hin auszudrucken.

Immerhin ist das Programm nicht ganz so einfach, wie es sein sollte. Das liegt an der Art und Weise, wie Daten geschrieben werden müssen. Daten für das Programm müssen wie folgt eingegeben werden: Tagesdatum, Querstrich(/), Monatsdatum, Querstrich, Jahreszahl. Falls der Computer die Querstriche nicht beachtet und die Daten nach ihrer Größe sortiert, würde er z. B. den 12. Dez. 1901 vor dem 1. Jan. 1999 einordnen (die Zahl 121201 ist größer als die Zahl 10199). Das Programm sollte die Daten besser in ihrer zeitlichen Abfolge schreiben können (die weiter zurückliegenden Daten würden dann als „kleinere Zahlen" behandelt), so daß es für den Benutzer einen Sinn ergibt. Auch die Tatsache, daß die Angabe von Tag und Monat einmal aus einer und dann wieder aus zwei Ziffern bestehen kann, ist ein Problem. Der Computer muß in der Lage sein, einstelligen Angaben von Tag und Monat eine Null voranzustellen (und natürlich auch wissen, wann eine Null nötig ist).

Die Routine dafür finden Sie zwischen den Zeilen 40 und 80. In Zeile 40 sollen Sie das Datum in der gewünschten Form eingeben. Zeile 50 ordnet dem Datum eine Zeichenkette zu, so daß es ein paar Zeilen weiter ausgedruckt werden kann. Zeile 60 prüft, ob der zweite Bestandteil des Daten-Strings ein Querstrich ist. Ist das Tagesdatum einstellig, wird ihm eine Null vorangestellt. Zeile 70 prüft, ob das fünfte Element des Daten-Strings ein Querstrich ist (wenn die Monatszahl nur aus einer Ziffer besteht). Wenn ja, dann wird der Monatszahl eine Null vorangstellt. In Zeile 80 wird das Datum neu geordnet, zuerst das Jahr, dann der Monat und zuletzt der Tag. (In Amerika müssen Sie übrigens den Befehl 4 TO 5 mit dem Befehl 1 TO 2 austauschen. Denn in den USA wird der Monat vor dem Tag angegeben, wenn das Datum nur mit Zahlen angegeben wird; entsprechend müßten Sie 25/12/84 in Zeile 40 in 12/25/84 ändern.)

Zeile 90 schreibt das Datum in seiner ursprünglichen Form oben rechts aus (weil wir B$ dem eingegebenen Datum zugeordnet haben). Geben Sie jetzt das „Ereignis" ein, das Sie zusammen mit dem Datum registriert haben wollen. Wie im Programm PRIVATAUSGABEN können Sie nun die Eingabe, so wie sie jetzt aussieht, akzeptieren oder ablehnen und von neuem eingeben.

Nehmen wir an, Sie sind mit der Eingabe einverstanden, dann ordnet das

Programm das Element des D$-String, der Ihren Kalender enthält, dem Datum und seinem Ereignis zu (dabei muß das Datum schon, für den Computer lesbar, aufbereitet sein).

Ab 220 sortiert die Routine den Inhalt des Terminkalenders und schreibt ihn in verständlicher Form aus. In Zeile 330 wird das Datum so geschrieben, daß Sie es gut lesen können. Unterhalb des Programms ist ein kurzer Probelauf angegeben. Vielleicht wollen Sie zur Übung noch ein Menü dazuhaben, um den Kalender – wie im Programm PRIVATAUSGABEN – speichern oder verbessern zu können. Wenn Sie mit der vorliegenden Form einverstanden sind, können Sie das Programm sofort speichern. Sie können dann später weitere Daten hinzufügen, die dann automatisch an der richtigen Stelle eingeordnet werden; vorausgesetzt, Sie starten das Programm mit GO TO 40 (statt mit RUN).

```
  10 REM TERMINKALENDER
  20 DIM d$(200,32)
  30 FOR d=1 TO 200
  40 INPUT "DATUM (Z.B. 25/12/84
) ";b$
  50 LET a$=b$
  60 IF a$(2)="/" THEN LET a$="0
"+a$
  70 IF a$(5)="/" THEN LET a$=a$
( TO 3)+"0"+a$(4 TO )
  80 LET a$=a$(7 TO 8)+a$(4 TO 5
)+a$(1 TO 2)
  90 PRINT AT 0,0;"DATUM: ";b$
 100 PRINT '"KURZTEXT (MAX. 22
ZEICHEN)"
 110 INPUT c$
 120 CLS
 130 PRINT AT 0,0;b$,"-";c$
 140 PRINT '"FALLS ALLES OKAY I
ST, BITTE "; INVERSE 1;"ENTER";
INVERSE 0,"ANDERENFALLS BITTE ";
 FLASH 1;"^E^"; FLASH 0," TIPPEN
!"
 150 INPUT e$: CLS
 160 IF e$<>"" THEN GO TO 40
 170 LET d$(d)=a$+c$
 180 PRINT '"TIPPEN SIE BITTE "
; INVERSE 1;"ENTER"; INVERSE 0;"
 ZUR EINGABE DES NAECHSTEN POSTE
N,"," ODER IRGENDEINEN ANDEREN B
UCHSTABEN MIT ENTER; UM DEN UEBE
RSICHTSPLAN ZU DRUCKEN!"
 190 INPUT e$: CLS
 200 IF e$="" THEN NEXT d
 210 PRINT PAPER 2; FLASH 1;"BIT
TE GEDULD...": PAUSE 100: CLS
 220 LET b=0
 230 LET g=d
 240 LET z=1
 250 LET b=z+1
 260 IF b>g THEN GO TO 330
 270 IF d$(b)>d$(z) THEN GO TO 2
90
 280 LET z=z+1: GO TO 250
 290 LET q$=d$(z)
```

```
300 LET d$(z)=d$(b)
310 LET d$(b)=q$
320 GO TO 280
330 PRINT d$(g)(5 TO 6);"/";d$(
g)(3 TO 4),"/";d$(g)(1 TO 2);"-"
;d$(g)(7 TO )
340 LET g=g-1
350 IF g>0 THEN GO TO 240
```

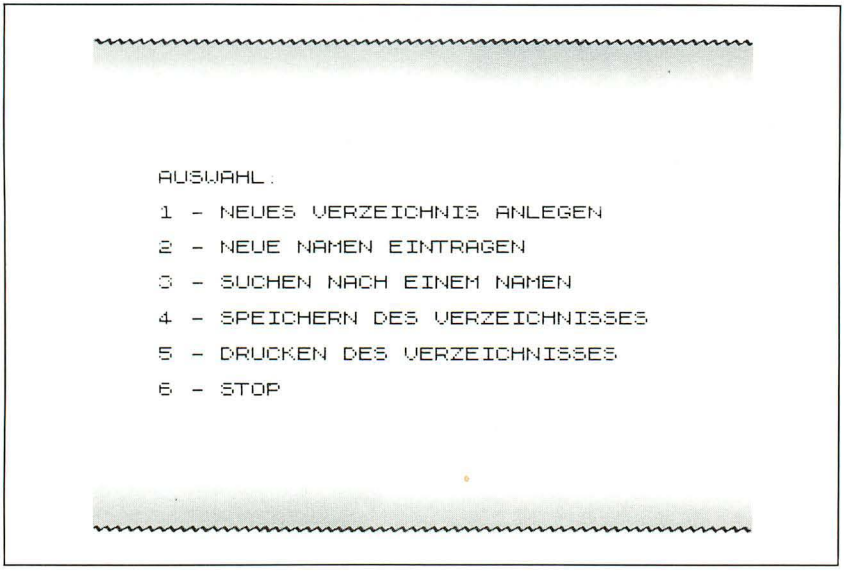
Probelauf: TERMINKALENDER

```
01/02/84-Zahnarzt
01/02/84-einkaufen
00/04/84-Urlaub
05/05/84-erster Arbeitstag
12/12/84-Weihnachtseinkauf
```

Telefonverzeichnis

Wahrscheinlich sehen Sie längst, was die Programme TERMINKALENDER, TELEFONVERZEICHNIS und DATENBANK miteinander verbindet. Im wesentlichen werden in allen drei Programmen Daten vom Anwender eingegeben, sortiert (bei DATENBANK gibt es mehrere Möglichkeiten) und dann in der richtigen Reihenfolge ausgedruckt bzw. auf eine Anfrage hin durchsucht.

Der Aufbau der drei Programme müßte Ihnen eigentlich mehr als einen Anhaltspunkt dafür geben, wie Sie selbst Programme zur Datenverwaltung und Sortierung für Ihre eigenen Problemstellungen schreiben können. Das Hauptmenü zeigt die verschiedenen Möglichkeiten:



Sie können im Prinzip 200 Zeilen mit bis zu 32 Zeichen eingeben. Wenn Sie sich aber mit Vornamen begnügen wollen, können Sie die Zahl der Eingaben sehr einfach erhöhen, indem Sie die Zeile 20 in DIM D$ (400,16) ändern. Sie geben den Namen und die Zahl ein und können noch einmal diese Angaben bestätigen, bevor sie in das Verzeichnis eingefügt werden.

Für die Eingabe eines neuen Namens drücken Sie ENTER und gelangen so zur Zeile 40, zur Eingabe des nächsten Satzes.

Wenn Sie das Verzeichnis sortieren wollen, drücken Sie irgendeine Taste (außer BREAK) und anschließend ENTER. Das Verzeichnis wird dann alphabetisch neu geordnet, entsprechend dem Anfangsbuchstaben des eingegebenen Namens (wenn das Verzeichnis Nachnamen aufführen soll, müssen Sie natürlich erst die Nachnamen eingeben). Jetzt werden Sie zum Menü zurückgeschickt. Wenn Sie die Nummer 3 drücken „Telefonnummer suchen", geht das Programm zur Zeile 500 und schreibt „Geben Sie den dazugehörigen Namen ein". Der Computer sucht dann im Verzeichnis (das nimmt verblüffend wenig Zeit in Anspruch) und drückt entweder die Nummer aus, wenn er sie findet, oder er meldet „Name nicht gefunden" (er sucht natürlich nur nach dem Namen, so wie Sie ihn eingegeben haben, mit den gleichen Zwischenräumen).

Mit diesem Programm können Sie wie im Programm PRIVATAUSGABEN das Verzeichnis auf Kassette speichern. Wenn es wieder geladen ist, können Sie es selbsttätig durchlaufen lassen (ohne daß es seinen Inhalt

verliert). Wenn Sie es mit Hand starten wollen, ohne den Inhalt zu löschen, fangen Sie mit GO TO 360 statt mit RUN an.

```
  10 REM TELEFON-VERZEICHNIS
  15 GO TO 360
  20 DIM d$(200,32)
  30 FOR d=1 TO 200
  40 INPUT "WELCHER NAME? ";b$
  90 PRINT AT 0,0;"NAME: ";b$
 110 INPUT "WELCHE TELEFON-NR. ?
";c$
 120 CLS
 130 PRINT AT 0,0;b$;"  ";c$
 140 PRINT '''"TIPPEN SIE "; INVE
RSE 1;"ENTER"; INVERSE 0;", WENN
 DIESES RICHTIG IST, ODER GEBEN
SIE "; FLASH 1;"^E^"; FLASH 0;"
UND DANN ENTER!"
 150 INPUT e$: CLS
 160 IF e$<>"" THEN GO TO 40
 170 LET d$(d)=b$+"  "+c$
 180 PRINT '''"TIPPEN SIE "; INVE
RSE 1;"ENTER"; INVERSE 0;" ZUR N
AECHSTEN EINTRAGUNG, ODER IRGEND
EINE TASTE UND DANN  ENTER ZUM S
ORTIEREN"
 190 INPUT e$: CLS
 200 IF e$="" THEN NEXT d
 210 PRINT PAPER 2; FLASH 1;"ER
SORTIERT..."
 215 POKE 23692,0
 220 LET b=0
 230 LET g=d
 240 LET z=1
 250 LET b=z+1
 260 IF b>g THEN GO TO 330
 270 IF d$(b)>d$(z) THEN GO TO 2
90
 280 LET z=z+1: GO TO 250
 290 LET q$=d$(z)
 300 LET d$(z)=d$(b)
 310 LET d$(b)=q$
 320 GO TO 280
 330 PRINT d$(g)
 340 LET g=g-1
 350 IF g>0 THEN GO TO 240
 360 PRINT '"AUSWAHL:"
 370 PRINT '"1 - NEUES VERZEICHN
IS ANLEGEN"
 380 PRINT '"2 - NEUEN NAMEN EIN
TRAGEN"
 390 PRINT '"3 - SUCHEN NACH EIN
EM NAMEN"
 400 PRINT '"4 - SPEICHERN DES V
ERZEICHNISSES"
 405 PRINT '"5 - DRUCKEN DES VER
ZEICHNISSES"
 410 PRINT '"6 - STOP"
 420 INPUT b: CLS
 430 IF b=1 THEN GO TO 20
 440 IF b=2 THEN NEXT d
 450 IF b=3 THEN GO TO 500
 460 IF b=4 THEN SAVE "VERZEICHN
IS"
 465 IF b=5 THEN FOR a=d TO 1 ST
EP -1: LPRINT d$(a): NEXT a
 470 IF b=6 THEN STOP
```

```
 480 GO TO 360
 500 PRINT '"EINGABE DES NAMENS
 ERFORDERLICH"
 510 INPUT a$: LET f=LEN a$
 520 PRINT '' FLASH 1; INK 1;"IC
H SUCHE NACH ";a$
 530 FOR a=1 TO d
 540 IF d$(a)( TO f)=a$ THEN PRI
NT ''d$(a)(f+1 TO ) : GO TO 360
 550 NEXT a
 560 PRINT '"NAME NICHT VORHANDE
N"
 570 GOTO 360
```

Probelauf: TELEFONVERZEICHNIS

```
KLEIBER 02324/00001
MEIER 0211/01234567
OTTO 0815/4711
ZITZEWITZ 09999/98765
```

Datenbank

Dieses Programm ist eigentlich eine Weiterentwicklung vom TELEFONVERZEICHNIS. Es soll ein Verzeichnis von einer Schallplattensammlung anlegen und speichern, einmal nach Künstlernamen und einmal nach Musiktiteln geordnet. Es ist in diesem Kapitel untergebracht, weil es etwas abgeändert auch bei einer Geschäftsdatei von Nutzen sein kann. Wie bei anderen menügelenkten Programmen, skizziert auch hier das Menü, wie Sie dabei vorgehen können.

1 – Anlegen einer neuen Datei
2 – Eintragen neuer Daten
3 – Ausdruck nach Künstlernamen
4 – Ausdruck nach Titeln
5 – Suche nach einem bestimmten Künstler
6 – Suche nach einem bestimmten Titel
7 – Speichern der Datenbank auf Kassette
8 – Programmende

Das Programm fordert auf, einen KÜNSTLER/KOMPONISTEN, dann einen TITEL und ein SACHGEBIET einzugeben. Dieses Sachgebiet bezieht sich auf Ihr eigenes Registriersystem. Es wird nicht zum Sortieren gebraucht, aber das Programm veranlaßt, daß es in sortierten Listen enthalten ist. So erhalten Sie bei der Suche nach einem Titel zusammen mit dem Titel den Künstler und das Sachgebiet. Wenn Sie dann nach einem Künstler suchen, erscheint ein Künstlername, ein Titel und das (entsprechende) Sachgebiet. Die Künstlerdatei ist im F$-Feld, und die Titeldatei im E$-Feld enthalten. Bei der Suche nach einem Künstler oder Titel vergleicht das Programm nur die ersten vier Buchstaben des Namens. Als Ausdruck sehen Sie einen Ausschnitt von fünf aufeinanderfolgenden Eintragungen der Datenbank, einmal nach Künstlernamen und einmal nach Titeln geordnet.

```
  10 REM DATENBANK
  20 PRINT '''"AUSWAHL:"
  25 PRINT '"1 - ANLEGEN EINER N
EUEN DATEI"
  30 PRINT '"2 - EINTRAGEN NEUER
 DATEN"
  40 PRINT '"3 - AUSDRUCK NACH K
UENSTLERNAMEN"
  50 PRINT '"4 - AUSDRUCK NACH T
ITELN"
  60 PRINT '"5 - SUCHE NACH EINE
M KUENSTLER"
  70 PRINT '"6 - SUCHE NACH EINE
M TITEL"
  80 PRINT '"7 - SPEICHERN DER D
ATEN AUF KASSETTE"
  90 PRINT '"8 - STOP"
 100 INPUT a
 110 CLS
 120 IF a=1 THEN GO SUB 210
 130 IF a=2 THEN NEXT j
 140 IF a=3 THEN GO SUB 630
 150 IF a=4 THEN GO SUB 670
 160 IF a=5 THEN GO SUB 710
 170 IF a=6 THEN GO SUB 800
 180 IF a=7 THEN SAVE "DATEN"
 190 IF a=8 THEN STOP
 200 GO TO 20
 210 DIM f$(130,24): DIM e$(130,
24)
 220 POKE 23692,0
 230 FOR j=1 TO 130: CLS
 240 PRINT AT 0,0; INK 2;"ZUM BE
ENDEN NEUER EINTRAGUNGEN  TIPPEN
 SIE BITTE  "Z"!"
 250 PRINT AT 10,10;"DATEN-NR";
FLASH 1;J
 260 INPUT "WELCHER KUENSTLERNAM
E? ";a$
 270 IF a$="Z" THEN GO TO 390
 280 PRINT 'a$
 290 INPUT "WELCHER TITEL? ";t$
 300 PRINT 't$
 310 INPUT "WELCHES SACHGEBIET?
";c$
 320 PRINT 'c$
```

```
 330 PRINT '"BITTE TIPPEN SIE ",
 INVERSE 1;"ENTER"; INVERSE 0;",
 WENN DIES RICHTIG IST; SONST IR
GENDEINE ANDERE TASTE UND DANN E
NTER!"
 340 INPUT z$
 350 IF z$<>"" THEN GO TO 240
 360 LET f$(j)=a$+":"+t$+":"+c$
 370 LET e$(j)=t$+":"+a$+":"+c$
 380 NEXT j
 390 CLS
 400 PRINT INK 1; FLASH 1,"BITTE
 ETWAS GELDULD BEIM SORT!"
 405 LET g=j: LET b=0
 410 LET z=1
 440 LET b=z+1
 450 IF b>g THEN LET g=g-1: IF g
>0 THEN GO TO 410
 455 IF g=0 THEN GO TO 520
 460 IF f$(b)<f$(z) THEN GO TO 4
80
 470 LET z=z+1: GO TO 440
 480 LET q$=f$(z)
 490 LET f$(z)=f$(b)
 500 LET f$(b)=q$
 510 GO TO 470
 520 LET g=j: LET b=0
 530 LET z=1
 540 LET b=z+1
 550 IF b>g THEN LET g=g-1: IF g
>0 THEN GO TO 530
 560 IF g=0 THEN CLS : GO TO 20
 570 IF e$(b)<e$(z) THEN GO TO 5
90
 580 LET z=z+1: GO TO 540
 590 LET q$=e$(z)
 600 LET e$(z)=e$(b)
 610 LET e$(b)=q$
 620 GO TO 580
 630 FOR m=1 TO j
 640 LPRINT f$(m)
 650 NEXT m
 660 RETURN
 670 FOR m=1 TO j
 680 LPRINT e$(m)
 690 NEXT m
 700 RETURN
 710 PRINT "WELCHER KUENSTLER? "
 720 INPUT n$: LET n$=n$+"    "
 730 FOR m=1 TO j
 740 IF f$(m)( TO 4)=n$( TO 4) T
HEN GO TO 770
 750 NEXT m
 760 PRINT "KUENSTLER FEHLT": PA
USE 200: RETURN
 770 PRINT f$(m)
 780 PAUSE 0
 790 RETURN
 800 PRINT "WELCHER TITEL? "
 810 INPUT n$: LET n$=n$+"    "
 820 FOR m=1 TO j
 830 IF e$(m)( TO 4)=n$( TO 4) T
HEN GO TO 860
 840 NEXT m
 850 PRINT "TITEL FEHLT": PAUSE
200: RETURN
 860 PRINT e$(m)
 870 PAUSE 0
 880 RETURN
```

Probeausdruck der DATENBANK:

```
BEATLES:HELP:008
CHOPIN:WALZER:34/3
MOZART:FANTASIE:K367
TRIO:DA DA DA:E3
UDO LINDENBERG:SONDERZUG:DE3

DA DA DA:TRIO:E3
FANTASIE:MOZART:K367
HELP:BEATLES:008
SONDERZUG:UDO LINDENBERG:DE3
WALZER:CHOPIN:34/3
```

Finanzplan/Verkaufsvorhersage

Wie schon eingangs erklärt, gibt dieses Programm eine grobe Vorstellung von den vorteilhaften Möglichkeiten, die von Kalkulationsprogrammen wie VisiCalc geboten werden. Unser Programm soll zunächst einmal die monatlichen Verkaufszahlen notieren sowie die Anzahl der Reklamationen, jede Veränderung beim Personal, kurz: jedes Ereignis von geschäftlicher Bedeutung, das ziemlich regelmäßig vorkommt und in regelmäßigen Abständen registriert werden muß. Von diesen Informationen ausgehend soll das Programm die monatlich zu erwartenden Gewinne im voraus berechnen, natürlich unter der Voraussetzung, daß die gegebenen Faktoren die gleichen bleiben.

Geben Sie eine 1 ein, wenn Sie eine „hardcopy" der Programm-Ergebnisse haben wollen (d. h. eine 1:1-Kopie des Druckers). Dann werden alle bedeutsamen Werte (natürlich ohne die Benutzer-Fragen) ausgedruckt (LPRINT) und zusätzlich auf dem Bildschirm gezeigt. Danach werden Sie gefragt: „Für wieviel Monate sind Angaben erhältlich?" Tauschen Sie gegebenenfalls Monate gegen Tage oder Jahre oder irgendeine andere Zeitspanne aus, die Sie brauchen. Dann sollen Sie pro Monat eine Angabe machen. Das Programm kann zur Zeit Eingaben für 19 Monate abarbeiten. Wenn Sie Daten für mehr als 19 Monate haben, tauschen Sie die Angabe „A−1" am Anfang der Zeile 100 gegen eine Null aus.

Der Spectrum wird dann die angenäherte Differenz zwischen zwei aufeinanderfolgenden Monaten berechnen. Dabei wird der zweite Monat mit dem ersten, der dritte mit dem zweiten Monat usw. verglichen. Die durchschnittliche, prozentuale Änderung wird anschließend ausgegeben.

Zur Vorhersage von Umsätzen müssen Sie die Zahl der Monate angeben, für die Sie Angaben vorausberechnet haben wollen. Es ist auch wichtig, ob diese Voraussage von der Angabe für den letzten Monat oder vom durchschnittlichen Monatsumsatz ausgehen soll.

Sobald das alles geklärt ist, stellt Ihnen das Menü folgende Punkte zur Wahl:

1 – die Vorausberechnung soll von neuem starten (d. h. Sie können für die Ausgangsangabe gleicherweise den monatlichen Durchschnittswert oder den letzten Monatsumsatz angeben, wobei der Zeitraum länger oder kürzer sein kann)

2 – noch einmal bis zur Ausgabe durchlaufen, jedoch ohne Eingabe von Zahlen (trotzdem müssen Sie die Fragen für die Vorausberechnung beantworten)

3 – Durchlauf des Programms von Anfang an (dadurch werden alle augenblicklich vorhandenen Daten gelöscht)

4 – Speichern auf Kassette (ein Rückgriff darauf ist möglich, wenn Sie nach dem Laden von Kassette Punkt 2 des Menüs wählen)

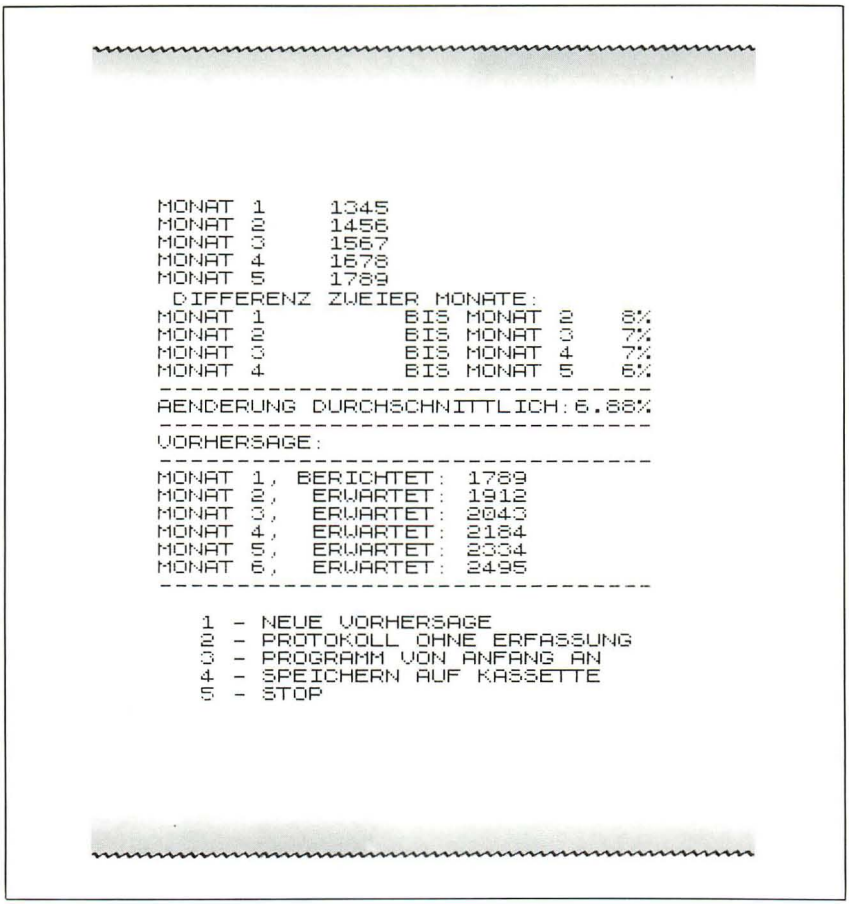
5 – Programmende

Wenn Sie nach einmaligem Durchlauf des Programms ohne Druckerausgabe einen Ausdruck wollen, stoppen Sie das Programm mit Punkt 5 des Menüs. Dann starten Sie wieder das Programm, und zwar mit GO TO 10 statt mit RUN, und wählen als nächstes Punkt 2 des Menüs. Jetzt haben Sie Ihre Angaben auf dem Drucker, ohne sie von neuem eingeben zu müssen.

```
  10 REM FINANZPLAN
        VERKAUFSVORHERSAGE
  15 GO SUB 490
  20 PRINT '''"FUER WIEVIEL MONAT
E SIND ANGABEN VORHANDEN?"
  30 INPUT m: LET sum=0
  40 IF m<2 THEN GO TO 30
  50 CLS: DIM a(m): DIM b(m)
  60 POKE 23692,0
  70 FOR a=1 TO m
  80 PRINT AT 20,0;"BETRAG IM MO
NAT "; FLASH 1; INK 2;a
  90 INPUT a(a)
 100 PRINT AT a-1,0; BRIGHT 1; P
APER 1; INK 7;" MONAT ";a;" ";TA
B (17-LEN STR$ a(a));a(a);"
  "
 105 IF Z=1 THEN LPRINT     "  MO
NAT ";a;" ";TAB (17-LEN STR$ a(a
));a(a);"          "
 110 LET sum=sum+a(a): NEXT a
 115 PRINT AT 20,0;"
 120 LET mi=sum/m: FOR b=2 TO m
 130 LET b(b)=(100-(a(b-1)*100/a
(b)))
 140 NEXT b
 145 PAUSE 100
 150 PRINT ''' FLASH 1; INK 1;"
DIFFERENZ ZWEIER MONATE: "
 155 IF z=1 THEN LPRINT " DIFFER
ENZ ZWEIER MONATE: "
 160 FOR a=2 TO m
 170 PRINT "MONAT "; INK 1;a-1,
INK 0;TAB 5;"BIS MONAT "; INK 1;
a; INK 0;"   ";INT (b(a)+.5);"%"
 175 IF z=1 THEN LPRINT "MONAT "
;a-1,"BIS MONAT ";a;"   ";INT (b(
a)+.5);"%"
 180 NEXT a
 185 PAUSE 100
 190 LET summe=0
 200 FOR a=2 TO m
 210 LET summe=summe+b(a)
 220 NEXT a
 230 LET mittelwert=INT (SUMME*1
00/(m-1))/100
 235 PRINT ' INK 2;"------------
--------------------"
 236 IF z=1 THEN LPRINT "-------
-------------------------"
 240 PRINT '"DURCHSCHNITTLICHE A
ENDERUNG:"; FLASH 1;mittelwert;
INK 0; FLASH 0;"%"
 245 IF Z=1 THEN LPRINT "DURCHSC
HNITTLICHE AENDERUNG:";mittelwer
t;"%"
 250 PAUSE 100
 255 PRINT ' INK 2;"------------
--------------------"
 256 IF z=1 THEN LPRINT "-------
-------------------------"
 260 PRINT '"VORHERSAGE:"
 265 IF z=1 THEN LPRINT "VORHERS
AGE:"
 270 INPUT "WIEVIEL MONATE SOLLE
N IM VORAUS BERECHNTET WERDEN?";
zahl
 280 PRINT '"LETZTER BERICHTSMON
AT WAR ";a(m)
```

```
 290 PRINT '"MITTELWERT PRO MONA
T WAR ";10*(INT mi/10)
 300 INPUT "SOLL SICH DIE VORHER
SAGE         1 - AUF DEN LETZTEN
MONAT  ODER 2 - AUF DEN MITTELWE
RT STUETZEN?";d
 305 PRINT ' INK 2;"------------
--------------------"
 310 LET e=(a(m) AND d=1)+((10*(
INT mi/10)) AND d=2)
 315 PRINT '"MONAT 1, BERICHTET:
";a(m)
 316 IF z=1 THEN LPRINT "MONAT 1
, BERICHTET: ";a(m)
 320 FOR a=2 TO zahl
 325 POKE 23692,0
 330 LET e=e+mittelwert*e/100
 340 PRINT "MONAT ";a;",  ERWART
ET: ";INT (e)
 342 IF z=1 THEN LPRINT "MONAT "
;a;",  ERWARTET: ";INT (e)
 345 NEXT a
 350 PAUSE 100: POKE 23692,0
 355 PRINT ' INK 2;"------------
--------------------"
 356 IF z=1 THEN LPRINT "-------
-------------------------"
 360 PRINT ' INK 2; INVERSE 1;"A
USWAHL:"
 370 PRINT 'TAB 3;"1 - NEUE VORH
ERSAGE"
 380 PRINT TAB 3;"2 - PROTOKOLL
OHNE ERFASSEN"
 390 PRINT TAB 3;"3 - PROGRAMM V
ON ANFANG AN"
 400 PRINT TAB 3;"4 - SPEICHERN
AUF KASSETTE"
 405 PRINT TAB 3;"5 - STOP"
 410 LET a$=INKEY$
 420 IF a$="" THEN GO TO 410
 430 IF a$="1" THEN GO TO 260
 440 IF a$="2" THEN GO TO 150
 450 IF a$="3" THEN RUN
 460 IF a$="4" THEN SAVE "VERKAU
F"
 470 IF a$="5" THEN STOP
 480 GOTO 360
 490 INPUT "PROTOKOLLDRUCK ERFOR
DERLICH ?    1 - JA
             2 - NEIN";z
 500 IF z<1 OR z>2 THEN GO TO 49
0
 510 RETURN
```

Druckausgabe: FINANZPLAN



```
MONAT 1     1345
MONAT 2     1456
MONAT 3     1567
MONAT 4     1678
MONAT 5     1789
 DIFFERENZ ZWEIER MONATE:
MONAT 1          BIS MONAT 2    8%
MONAT 2          BIS MONAT 3    7%
MONAT 3          BIS MONAT 4    7%
MONAT 4          BIS MONAT 5    6%
--------------------------------
AENDERUNG DURCHSCHNITTLICH:6.88%
--------------------------------
VORHERSAGE:
--------------------------------
MONAT 1, BERICHTET:  1789
MONAT 2,  ERWARTET:  1912
MONAT 3,  ERWARTET:  2043
MONAT 4,  ERWARTET:  2184
MONAT 5,  ERWARTET:  2334
MONAT 6,  ERWARTET:  2495
--------------------------------

   1 - NEUE VORHERSAGE
   2 - PROTOKOLL OHNE ERFASSUNG
   3 - PROGRAMM VON ANFANG AN
   4 - SPEICHERN AUF KASSETTE
   5 - STOP
```

Vorschläge für weitere Literatur zu diesem Thema:

Struble, George: Business Information Processing with BASIC, Addison-Wesley Publishing Company, USA, 1980

Sternberg, Charles D.: BASIC Computer Programs for Business, Hayden Book Company, Inc., USA, 1981

Gilder, Jules H.: BASIC Computer Programs in Science and Engineering, Hayden Book Company, Inc., USA, 1981

Poole, Lou und Borchers, Mary: Some Common BASIC Programs, Osborne/McGraw-Hill, USA, 1981

Miller, Alan R.: BASIC Programme. Mathematik – Informatik – Statistik, SYBEX Verlag GmbH, Düsseldorf (erscheint Juni 1983)

Bui, X. T.: Executive Planning with BASIC, SYBEX Inc., USA, 1982

Hergert, D.: BASIC for Business, SYBEX Inc., USA, 1982

Kapitel 6

Der Spectrum als Lehr- und Lernmittel

Dieses Kapitel befaßt sich mit dem Einsatz des Spectrum im Unterricht. Es soll dem Lehrer Hilfestellung bei der Entwicklung von Lehrprogrammen geben; es soll gegebenenfalls auch den Eltern helfen. Nicht zuletzt wird auch derjenige, der in der Ausbildung steht, einen Gewinn von diesem Kapitel haben, sei es, daß er hinter die Tricks von Lehrprogrammen kommt, oder sei es, daß er sich selbst oder seinen Mitschülern einige hilfreiche Programme schreibt.

Im kommerziellen Bereich krankt Unterrichtssoftware sowohl beim Spectrum als auch bei anderen Computern an ein bis zwei elementaren Fehlern: Entweder sie ist zu einfach bzw. zu allgemein, so daß jeder nur einen minimalen Gewinn davon hat, oder aber sie ist so spezialisiert, daß sie wiederum nur für ganz wenige tauglich ist. Nutzen wir jedoch den Computer, um uns selbst eigene Lernprogramme zu schreiben, können wir beide Extreme vermeiden. Wir können die Software exakt auf unsere eigenen Bedürfnisse abstimmen und vornehmlich die Teile behandeln, welche der besonderen Aufmerksamkeit bedürfen. Die in diesem Kapitel behandelten Routinen und Programme werden so aufgestellt, daß sie leicht an die eigenen Bedürfnisse angepaßt werden können.

Das Kapitel konzentriert sich auf Programme, die praktische Übungen in elementaren Techniken bieten, so z. B. das Arbeiten mit Zahlen oder das Erwerben von Kenntnissen in Französisch. Es gibt viele andere Arten von Lehrprogrammen, die neue Lehrinhalte vermitteln oder die vielleicht bei der zeitraubenden Laborarbeit eingesetzt werden könnten, z. B. Mischversuche mit Chemikalien bei vorgegebenen Mengen. Wir werden sehen, daß eines oder mehrere der hier angegebenen Programme uns in die Lage versetzen werden, Programme nach eigenen Vorstellungen anzupacken.

Lehrprogramme müssen nicht langweilig sein. Manche Lehrprogramme kann man sogar ganz bewußt als Spielprogramme schreiben. Spiele mit Unterrichtsinhalten verlangen von den Spielern, eigene Kenntnisse zu

entwickeln und einzusetzen, um den Wettbewerb mit anderen zu gewinnen. Richtig angewendet sollte der Erfolg denjenigen zukommen, die ihre Kenntnisse am wirkungsvollsten einsetzen. Sicherlich werden auch Zufallsfaktoren eine gewisse Rolle spielen, aber sie sollen nicht spielentscheidend sein.

Programme, die neue Lehrinhalte vermitteln, können u. U. sehr lang sein, da der Programmierer alle nur denkbaren Fehler und Irrwege auffangen und korrigieren sowie Anleitungen dafür vorsehen muß. Außerdem muß der Inhalt unter verschiedenen Perspektiven dargeboten werden, damit der Anwender, welcher die Erklärungen nicht gleich beim ersten Mal versteht, möglicherweise doch Erfolg hat, wenn die Informationen noch auf eine andere Weise dargeboten werden.

Das klingt alles sehr anstrengend, ja fast unmöglich. Vielleicht ist es eine ganz gute Hilfe, zu wissen, daß der oder die Schüler in den meisten Fällen immer noch ein begleitendes Buch zur Hand haben. Dann kann das Programm mehr als intelligenter Führer zum Buch und weniger als ein Alleinunterhalter gesehen werden. Das Programm könnte z. B. sagen: „Nun, nachdem Du gezeigt hast, daß Du den Lamaismus und den aus der Mahayana-Form des Buddhismus abgeleiteten Ritus verstanden hast, möchte ich Dich auf Seite 26 des Textbuches verweisen. Dort kannst Du lesen, wie der Ritus nach Tibet gelangt ist. Wenn Du es gelesen hast, wende dich wieder dem Computer zu! Ich werde dir dann einige Fragen darüber stellen!" Danach kann mit den Fragen der Inhalt von Seite 26 geprüft werden. Hat der Schüler einiges davon verstanden, so kann er zu einem bestimmten Punkt auf dieser Seite oder auf eine weitere Ausführung über das Thema irgendwo anders im Buch verwiesen werden. Wenn man das Programm so aufbaut, spart man wertvolle Zeit und braucht nicht endlose PRINT-Befehle zu schreiben, um alle erdenklichen Fälle abzudecken. Dies bedeutet auch, daß man den Spectrum dazu verwenden kann, Gegenstände zu unterrichten, die anderenfalls für den Computereinsatz zu komplex sind.

Oft haben die Schüler Probleme, geeignete Informationen zu finden. Der Spectrum kann dazu verwendet werden, die Information selbst zu speichern, welche dann über Menü-Zugriffe wie beim Bildschirmtext zugänglich ist. Er kann aber gelegentlich auch dazu dienen, geeignete Quellenhinweise zu geben. Abgesehen von den unmittelbaren Vorteilen für den Schüler dürfte dies auch den Eindruck vermitteln, wie wichtig Informationsverarbeitung in der modernen Gesellschaft ist. Wenn eine Schule, oder wie in diesem Fall ein Haus, keinen Zugriff auf Bildschirmtext hat, kann man versuchen, dieses System mit einer begrenzten Zahl von Wahlmöglichkeiten nachzubilden. Hierdurch wird der Schüler mit dem Umgang mit Menü-Techniken und dem Zugriff auf Informationen, die nach einer Art Baumstruktur mit Querbeziehungen gespeichert sind, vertraut gemacht.

Auf der anderen Seite helfen Simulationen, Untersuchungen anzustellen, die man im täglichen Leben nicht nachvollziehen kann. Der Spectrum kann ein mathematisches Modell benutzen, um den Einfluß von Systemänderungen auf das Ergebnis vorherzusagen. Man möchte beispielsweise feststellen, welchen Einfluß der jeweilige Grad der Verschmutzung auf Fische, Insekten und Pflanzen hat. Im wirtschaftlichen Bereich erlaubt die Simulation, Veränderungen des Kapitals und dessen Einfluß auf den Beschäftigungsgrad zu studieren.

Bei einer chemischen Simulation kann man den Katalysator, die Temperatur und den Druck verändern, um z. B. das Ausbringen an Schwefeldioxyd in einer Fabrik, die Schwefelsäure herstellt, zu optimieren. Natürlich kann eine Simulation nur dann realitätsbezogen sein, wenn das mathematische Modell eine genaue Wiedergabe des Systems liefert. Selbst wenn es nicht ganz genau ist, so kann es doch zumindest lehrreich sein, solange nur der Schüler über die Vereinfachungen informiert wird.

Am Ende dieses Kapitels werden einige zur Unterrichtssoftware typische Wege, die sich als erfolgreich für andere erwiesen haben, aufgezeigt und eine Idee davon vermittelt, welche Dinge leicht in Lehrprogramme übertragen werden können. Zuerst jedoch sollen Programme (z. T. von Jeff Warren) vorgestellt werden, welche die unterschiedlichen Schritte zur Herstellung derartiger Software zeigen. Selbst wenn die Programme in dieser Form nicht unmittelbar benutzt werden können, so dürften sie doch als Vorlage dienen, um eigene Programme herzustellen.

Viele Lehrprogramme benutzen die RND-Funktion, um Zufallswerte zu bilden oder um Fragen in zufälliger Folge auszuwählen. Hierdurch soll vermieden werden, daß der Schüler mit einer vorhersehbaren Folge von Fragen gelöchert wird. Unser erstes Programm erzeugt einfache Divisionsaufgaben mit Zufallswerten. Es läßt sich jedoch leicht abwandeln, um andere mathematische Techniken zu üben.

Zeile 5 verwandelt das frei definierbare graphische Symbol D in ein Divisionszeichen, wie es gewöhnlich in der Schule verwendet wird. Danach wird der Punktestand S auf 0 gesetzt. Das Programm stellt 20 Fragen. Hierzu wird die FOR/NEXT-Schleife mit Q als Nr. der Frage verwendet (Zeilen 20–120). Die Fragen benutzen die Form „Wieviel ist 32 dividiert durch 4?". Der Spectrum setzt hierfür x/y. Zeile 30 liefert einen Wert für Y zwischen 2 und 12 und Zeile 40 für X einen Wert, der als Ergebnis einen ganzzahligen Wert zwischen 1 und 12 liefert. Zeile 50 zeigt die Aufgabe und nimmt die Antwort mit INPUT a entgegen. Zeile 50 wählt je nach Ergebnis die Zeilen 60 und 80 zur Angabe von Korrekturen oder Zeile 90, die den Schüler lobt und außerdem das Punktekonto enthält. Zur Erinnerung: Um den Grafik-Cursor zu erhalten, muß in Zeile 80 gleichzeitig CAPS SHIFT und Taste 9 gedrückt werden. Während des Programmlaufs ändert sich das D in einen Schrägstrich.

Zeile 100 gibt dem Anwender die Möglichkeit, mit dem nächsten Problem fortzusetzen, sobald er soweit ist. Wie schnell der Schüler die Information vom Bildschirm aufnimmt, ist sehr unterschiedlich. Eine feste Pause ist daher nicht so gut. Sicherlich könnte der Befehl PAUSE benutzt werden. Dabei müßte jedoch eine ausreichende Zeitspanne berücksichtigt werden, da sonst der Text von der Bildfläche vor dem Lesen verschwindet und der Leser keine Chance hätte, den Text nochmals wiederzusehen. Der Benutzer sollte niemals durch feste FOR/NEXT-Warteschleifen gezwungen werden, auf den nächsten Teil zu warten, da dies auf Dauer sehr frustrierend wirkt. Zeile 100 erlaubt, das Programm mit der Eingabe S und ENTER zu stoppen. Die letzte Zeile schließlich zeigt zum Schluß den Punktestand.

DIVISIONSÜBUNGEN

```
    5 FOR f=0 TO 7: READ a: POKE
USR "d"+f,a: NEXT f: DATA 0,16,0
,254,0,16,0,0
   10 LET s=0
   20 FOR q=1 TO 20
   30 LET y=2+INT (RND*11)
   40 LET x=y*INT (RND*12+1)
   50 PRINT AT 9,0; PAPER 6;"FRAG
E ";q'': PRINT "WIEVIEL IST ";x;
" DIVIDIERT DURCH ";y;"?": INPUT
 a: IF a=x/y THEN GO TO 90
   60 PRINT PAPER 6;''"FALSCH. DI
E ANTWORT IST NICHT ";a
   80 PRINT ''x;" d ";y;" = ";x/y
: PRINT PAPER 6;'"(ES IST NAEMLI
CH ";y;" * ";x/y;" = ";x;")": GO
 TO 100
   90 PRINT ''"GUT, ";a;" IST RIC
HTIG.": LET s=s+1
  100 INPUT "ZUR FORTSETZUNG BITT
E ENTER EIN-GEBEN";c$: IF c$="s"
 THEN STOP
  110 CLS
  120 NEXT q
  130 PRINT AT 0,4;"SIE ERREICHTE
N ";s;" PUNKTE VON 20 MOEGLICHEN
!"
```

Der Schwierigkeitsgrad der Aufgaben muß auf die Fähigkeiten des Schülers abgestimmt sein. Der Rechner sollte diese Aufgaben nicht etwa in der Art: „Wieviel ist 19 geteilt durch 7?“ stellen, wenn der Schüler nichts von Divisionsresten und Dezimalzahlen versteht.

Das Programm kann besser auf den Anwender eingestellt werden, wenn der Punktestand den Schwierigkeitsgrad der Aufgaben berücksichtigt. Eine Möglichkeit hierzu wäre beispielsweise, die Zeile 30 abzuändern in:

```
30 LET y = 2 + INT (RND* (3+ (s>=5) *4 + (s>=10) *4))
```

Y erhält Werte im Bereich von 2 bis 4, solange der Punktestand unter 5 liegt, denn die logischen Ausdrücke (s>=5) und (s>=10) sind beide nicht erfüllt, also wertmäßig null. Wenn der Punktestand über oder bei 5 liegt, ist der Ausdruck (s>=5) wahr und hat dann den Wert 1. Hierdurch wird y Werte zwischen 2 und 8 annehmen. Wenn schließlich der Punktestand den Wert 10 erreicht hat, erhält y Werte bis maximal 12. Entsprechend läßt sich auch Zeile 40 ändern:

```
40  LET x = y* INT (RND* (5+ (s>=5) *7) +1)
```

Der Vorteil dieser Art von Programmkontrollen liegt darin, daß der Übergang zu schwierigeren Aufgaben langsam erfolgt. Die Übungen werden dadurch einfacher und zugleich spannender.

Das Programm wurde zwar als Divisionsübung geschrieben, läßt sich aber leicht für andere Rechenarten abändern. Wir können den zweiten Teil von Zeile 50 ändern in:

„Wieviel ist 1/“;y;“ von “;x;“?“

Dies liefert Aufgaben der Art „Wieviel ist 1/4 von 32?“ zum Üben von Brüchen. Zeile 80 muß dann entsprechend geändert werden. Es ist leicht, Programmversionen zu erstellen, die die Addition und Subtraktion (Achtung: Die erste Zahl muß dann größer sein als die zweite, es sei denn, der Schüler kennt sich mit negativen Zahlen aus) sowie die Multiplikation behandeln.

Etliche Programmierer ziehen den persönlichen Stil vor und sprechen den Benutzer mit seinem Namen an. Die Maschine erscheint hierdurch etwas menschlicher. Alles was man hierzu braucht, ist ein Unterprogramm, welches etwa in folgender Weise abläuft:

Spectrum: „Hallo! Wie heißt Du?“
Schüler: „Sabine“
Spectrum: „Gut, Sabine! Ich hoffe, Du bist bereit, einige Fragen zur Anatomie der Ratte zu beantworten!“

Dieses kann mit einer Liste von „lobenden Redewendungen“ fortgeführt werden, welche wahlweise bei einer korrekten Antwort verwendet werden:

„Das ist richtig, Sabine! . . . stimmt!“

oder

„Gut gemacht, Sabine! . . . war richtig!“

Diese Vorgehensweise kommt denjenigen entgegen, die sich für sprechende Computer oder Science-fiction-Roboter begeistern.

Der Spectrum kann auch zum Wettkampf anspornen, indem er das je-

weils beste Ergebnis speichert. Zusätzlich kann noch die kürzeste Antwortzeit festgehalten werden. Dies hilft besonders dann, wenn ein Schüler die eigene Leistung verbessern möchte. Es kann jedoch auf die langsameren Schüler in einer Gruppe entmutigend wirken. Wir sollten daher sehr vorsichtig damit umgehen.

Im nächsten Programm, dem Rechen-Quiz, wird der Name des Schülers vom Computer verwendet, und die ganze Ansprache ist wesentlich freundlicher als zuvor beim Divisionstest. Außerdem erlaubt es noch dem Schüler, zwischen den verschiedenen Aufgabenbereichen (Addition, Subtraktion, Multiplikation, Division) zu wählen.

```
  10 REM RECHENQUIZ
  30 PRINT '''"HALLO !          "''
"WIR WOLLEN EIN PAAR SUMMEN BILD
EN."
  40 PAUSE 200
  50 CLS
  60 PRINT '''"WIE HEISSEN SIE?"
  70 INPUT a$
  80 CLS
  90 PRINT ''"ES FREUT MICH, SIE
 ZU SEHEN, "; FLASH 1; BRIGHT 1;
INK 1;a$
 100 PRINT '''"DRUECKEN SIE EINE
 TASTE, WENN   SIE STARTEN MOECH
TEN."
 110 PAUSE 0: LET punkte=0
 120 CLS
 130 PRINT '"NUN, ";a$;", ICH KA
NN FRAGEN STELLEN UEBER:"
 140 PRINT '"1 - ADDITION;"
 150 PRINT '"2 - SUBTRAKTION;"
 160 PRINT '"3 - MULTIPLIKATION
; ODER"
 170 PRINT '"4 - DIVISION."
 180 PRINT ''"WAEHLEN SIE EINE N
UMMER VON "; FLASH 1;"1 BIS 4"'F
LASH 0;" UM DIE RECHENART ZU WAE
HLEN, IN DER SIE SICH UEBEN WOLL
EN"
 190 INPUT b$: LET b=VAL b$
 200 IF b<1 OR b>4 THEN GO TO 19
0
 205 CLS
 210 PRINT ''"IN ORDNUNG, ";a$;"
, WIR UEBEN ETWAS ";
 220 PRINT INK 2; FLASH 1; ("ADDI
TION" AND b=1)+("SUBTRAKTION" AN
D b=2)+("MULTIPLIKATION" AND b=3
)+("DIVISION" AND b=4)
 230 PAUSE 300
 240 CLS
 250 PRINT '''"TIPPEN SIE '1', W
ENN SIE LEICHTE"'"ODER '5', WENN
 SIE SCHWIERIGE AUFGABEN MOECHTE
N."'''"SIE KOENNEN AUCH '2','3'
ODER '4' WAEHLEN, WENN SIE NICHT
 ZU EINFACHE, ABER AUCH NICHT ZU
 SCHWIERIGE AUFGABEN WUENSCHEN."
 260 INPUT c$: LET c=VAL c$
 270 IF c<1 OR c>5 THEN GO TO 26
0
```

```
 275 CLS
 280 LET d$=("+" AND b=1)+("-" A
ND b=2)+("*" AND b=3)+("/" AND b
=4)
 290 FOR e=1 TO 10
 300 LET a=INT (RND*(10+c))+c
 310 LET b=INT (RND*(10+4*c))+c
 320 LET g$=STR$ a+d$+STR$ b
 325 IF ABS (INT VAL g$)<>VAL g$
 THEN GO TO 300
 330 PRINT "NUN, ";a$;", DAS IST
";TAB 8;"FRAGE NR. ";e
 340 PRINT '''"WAS ERGIBT     ";g
$;"?"
 350 PAUSE 50
 360 PRINT '''"VERSUCHEN SIE ES
BITTE JETZT,UND TIPPEN SIE IHRE
ANTWORT EIN."
 370 INPUT f
 375 PRINT ''
 380 IF f=VAL g$ THEN BEEP .08,f
+punkte: BEEP 1,2*(f+punkte): PR
INT FLASH 1; INK RND*7; PAPER 9;
"GUT GEMACHT, ";a$: LET punkte=p
unkte+1
 390 IF f<>VAL g$ THEN PRINT "ES
 TUT MIR LEID, ";a$;", ABER DIE
ANTWORT IST ";VAL g$
 400 PAUSE 50
 410 PRINT '''"IHRE PUNKTZAHL BE
TRAEGT NUN";INK 1; FLASH 1;punkt
e; INK 0;FLASH 0;" VON ";e
 420 IF e<10 THEN PRINT '''"DRUE
CKEN SIE BITTE EINE TASTE   FUER
 DIE NAECHSTE FRAGE.": PAUSE 0:
CLS: NEXT e
 430 PAUSE 100
 440 CLS
 450 PRINT ''"DAS QUIZ IST ZUEND
E,";A$;"."
 460 PAUSE 100
 465 PRINT '''"IHRE PUNKTZAHL BE
TRUG ";INK 1; FLASH 1;punkte;INK
 0; FLASH 0;" VON ";e;TAB 8; FLA
SH 1; BRIGHT 1; INK 2;punkte *10
0/e;"%"
 470 PRINT ''"MOECHTEN SIE EIN N
EUES SPIEL?"''"TIPPEN SIE ENTER,
WENN SIE FORTSETZEN MOECHTEN, OD
ER IRGENDEINE ANDERE TASTE, WENN
 SIE AUFHOEREN MOECHTEN!"
 480 INPUT y$
 490 IF y$="" THEN GO TO 100
 500 PRINT '''"ES HAT MIR SPASS
GEMACHT, MIT IHNEN, ";a$;" ZU RE
CHNEN!"
 520 print ''' FLASH 1;"BIS ZUM
NAECHSTEN MAL!"
 530 STOP
```

Wie man während des Programmablaufes feststellt, werden Farbe und Blinken benutzt, um bestimmte Dinge hervorzuheben (z. B. beim ersten Mal den Namen des Schülers). Der Dialog wird von diesem Programm wie folgt durchgeführt:

„Hallo, . . ."
„Wir wollen ein paar Rechnungen probieren."
„Wie heißt Du? (Der Schüler gibt seinen Namen ein)"
„Guten Tag (Blinken, ..., blaue Schrift), Sabine"
„Tippe irgendeine Taste, wenn's losgehen soll."

Pause 0 wird hier (Zeile 110) benutzt, um mit dem Bild so lange zu warten, bis eine Taste eingetippt wurde. Die Variable PUNKTE (welche den Punktestand festhält) wird auf Null gesetzt. Der Bildschirm wird gelöscht (Zeile 120). Der Computer greift den Dialog wieder auf mit:

„Sabine: Ich bin bereit, dir Aufgaben zu stellen über:

1 – Addition
2 – Subtraktion
3 – Multiplikation oder
4 – Division.

Wähle eine Zahl zwischen (FLASH an) 1 bis 4 (FLASH aus), um mir mitzuteilen, welchen Aufgabentyp Du versuchen möchtest."

Der Schüler gibt eine Zeile an. Diese wird, wie man in Zeile 190 sieht, als String angenommen und dann (mittels VAL) in eine Zahl umgeformt.

Übrigens: Programme müssen so geschrieben werden, daß sie bei fehlerhaften Eingaben möglichst nicht abstürzen. Dies wird von dem ersten Programm dieses Kapitels nicht voll abgefangen. Die Eingabe eines Buchstabens statt einer Ziffer könnte das Programm veranlassen, mit einer Fehlermeldung abzubrechen. Der Benutzer wüßte nicht, was zu tun ist. Dieses Problem läßt sich vermeiden, wenn man die Eingabe mit einer Stringvariablen durchführt.

Man kann die CODE-Funktion verwenden, um sicherzustellen, daß die Eingabe, sofern erforderlich, numerisch ist. Danach kann man den String mittels VAL in eine Zahl umwandeln. Mit den folgenden Zeilen soll eine Ziffer erfaßt werden:

```
10  INPUT n$
20  IF LEN n$ <> 1 THEN PRINT
    "Bitte nur eine Ziffer!": GOTO 10
30  IF CODE n$ < 48 OR CODE n$> 57
    THEN PRINT "Bitte nur Zahlen!": GOTO 10
40  LET n=VAL n$: PRINT "Das war's."
```

Wie wir sehen, kann das Programm nur mit einer einstelligen Zahl beendet werden (Anmerkung: Die Ziffern 0 bis 9 belegen in der ASCII-Tabelle die Plätze 48 bis 57 und werden daher im Rechner mit ihrer Platznummer notiert).

Bei mehrstelligen Zahlen muß man die Eingabe in Einzelzeichen zerlegen und die Zeichen des Strings nacheinander überprüfen. Stattdessen kann man auch den Befehl INKEY$ verwenden, um die Zahl ziffernweise zu erfassen. Mit dem Code INKEY$=13 (welches die Platznummer für das ENTER-Zeichen ist) wird dann die Eingabe als beendet angesehen.

Nun zurück zu unserem Rechen-Quiz. Der Computer hat den Benutzer nach einer Zahl von 1–4 gefragt, um den Aufgabentyp auszuwählen. Der Dialog setzt, falls die 2 gewählt wurde, folgendermaßen fort:

„Okay, Sabine! Wir wollen ein paar Subtraktionen probieren."

Man beachte, wie der Spectrum in Zeile 220 den logischen Ausdruck AND berechnet und je nach Ergebnis für B eines der vier Wörter (Addition, Subtraktion, Multiplikation oder Division) auswählt. In Zeile 230 befindet sich eine Pause. Der Bildschirm wird gelöscht und der Spectrum fährt fort: „Gib eine 1 ein, falls du leichte Berechnungen, oder tippe eine 5, wenn Du die schwierigeren Aufgabenstellungen wünschst. Du kannst aber auch eine 2, 3 oder 4 tippen, um mittelschwere Aufgaben zu erhalten." Hierdurch erhält der Benutzer ein Gefühl der Kontrolle über seinen Computer und gewinnt den Eindruck, daß er mit einem intelligenten Roboter spricht. Die Eingabe wird als String akzeptiert und anschließend in eine Zahl umgewandelt. Der Bildschirm wird gelöscht. Zeile 280 nutzt erneut die logische Berechnung AND, um die arithmetische Operation (+, −, * oder /) auszuwählen und dem Benutzer in der Aufgabenstellung vorzulegen.

Mit Zeile 290 startet eine Schleife, in der zehn Fragen an den Schüler gerichtet werden; mit Zeilen 300 und 310 werden Zahlen ausgesucht, deren Größe vom in Zeile 260 gewählten Schwierigkeitsgrad (Variable C) des Problems abhängt. Zeile 320 packt die Zahlen und den Operator in einem einzigen String zusammen; Zeile 325 prüft, ob das Ergebnis ganzzahlig (dies könnte bei einer Division passieren) und positiv ist (bei Subtraktionen). Die Kontrollzeile kann weggelassen werden, wenn Dezimalbrüche und negative Zahlen erlaubt sein sollen.

Zeile 330 zeigt die Aufgabenstellung und spricht dabei den Schüler direkt an:

„Nun, Sabine! Jetzt kommt Frage 4."
„Wieviel ist 7 + 3?"
(In Zeile 350 wird eine kurze Pause gemacht)
Danach sagt der Spectrum:
„Überlege jetzt und gib dann Deine Antwort!"

Zeile 380 prüft die Antwort und lobt den Schüler mit mehreren Tönen, falls die Antwort richtig war. Die Tonhöhe richtet sich nach der Zahl der bisherigen, richtigen Antworten. Die Worte „Gut gemacht, Sabine" erscheinen auf dem Bildschirm mit zufällig gewähltem, blinkenden Farb-

ton. Hierbei wird Shift 9 verwendet, um sicherzustellen, daß jetzt keine blinkenden Wörter und keine Töne erzeugt werden. Bei einer richtigen Antwort erhält der Schüler also eine bessere „Belohnung" als bei einer falschen Antwort.

Es folgt eine weitere kurze PAUSE. Danach sagt der Spectrum:

„Dein Punktestand ist nunmehr (INK 1, FLASH) 1
(FLASH aus, INK normal) von 3."

Vor der zehnten Frage sagt der Spectrum:

„Tippe irgendeine Taste für die nächste Aufgabenstellung."

Dieses Warten auf den Tastendruck erlaubt dem Schüler, in der eigenen Geschwindigkeit voranzuschreiten.

Am Ende der zehnten Frage, nach einer PAUSE, erscheint auf dem Schirm:
„Das bringt uns zum Ende des Quiz, Sabine." PAUSE
„Dein Punktestand war (INK 1, FLASH) 8
(INK normal, FLASH aus) von 10, (INK 2, FLASH 1) also 80%.
„Möchtest Du eine neue Runde machen? Wenn ja, tippe ENTER. Anderenfalls tippe irgendeine Taste, um das Programm zu beenden."

Obwohl es eine ziemlich lange Zeit in Anspruch nimmt, ein solches Dialog-Programm zu schreiben, macht sich dieser zusätzliche Aufwand für den Schüler sicherlich mehr als bezahlt.

Falls der Schüler einen weiteren Test machen möchte, beginnt der Computer nicht wieder ganz von vorn (RUN), sondern springt zur Zeile 100. Hierdurch wird vermieden, daß der Schüler seinen Namen noch einmal eingeben muß. Der Eindruck, daß der Computer ein intelligenter, hilfreicher Roboter ist, bliebt dadurch erhalten. Wenn der Schüler auf die Frage nach einem weiteren Test nicht mit ENTER antwortet, sagt der Computer:

„Das Rechnen mit Dir, Sabine, ging ganz gut (FLASH an). Bis zum nächsten Mal."

Obwohl diese Erläuterungen wesentlich länger geraten sind als das eigentliche Programm, sehen wir, wie wirkungsvoll man ein Programm mit Dialog-Texten gestalten kann. Der Vergleich der Ausgabe des Arithmetik-Quiz mit dem Divisionsprogramm zeigt, daß das zweite Programm wesentlich benutzerfreundlicher ist als das erste.

Programme werden dadurch benutzerfreundlicher, daß die Meldung „Gut gemacht" etwas ausgestaltet wird. Viele Schüler haben solchen Spaß an dieser zusätzlichen Motivation, daß sie Schulübungen mit einer Art Spiel verbinden oder mit optisch interessanten Belohnungen anreichern. Wir nehmen uns jetzt das erste Programn dieses Kapitels vor und statten es mit Farbe und Ton aus.

Im folgenden Programm sehen Sie jetzt oben auf dem Bildschirm einen Fluß mit einer unvollständigen Brücke. Auf dem linken Flußufer steht ein Panzer. Bei jeder richtigen Antwort wird die Brücke ein Stück weitergebaut. Sind alle 20 Antworten richtig, dann wird eine Siegeshymne gespielt, der Panzer überquert die Brücke und feuert.

Zuerst müssen wir Zeile 5 des DIVISIONS-Programms umändern in:

```
5 GO TO 150
```

Dann fügen wir am Ende der Zeile 90 den Befehl an:

```
:GO SUB 250
```

und ändern Zeile 110 in:

```
110 PRINT AT 8,0;: FOR n=1 TO 11: PRINT b$:NEXT n
```

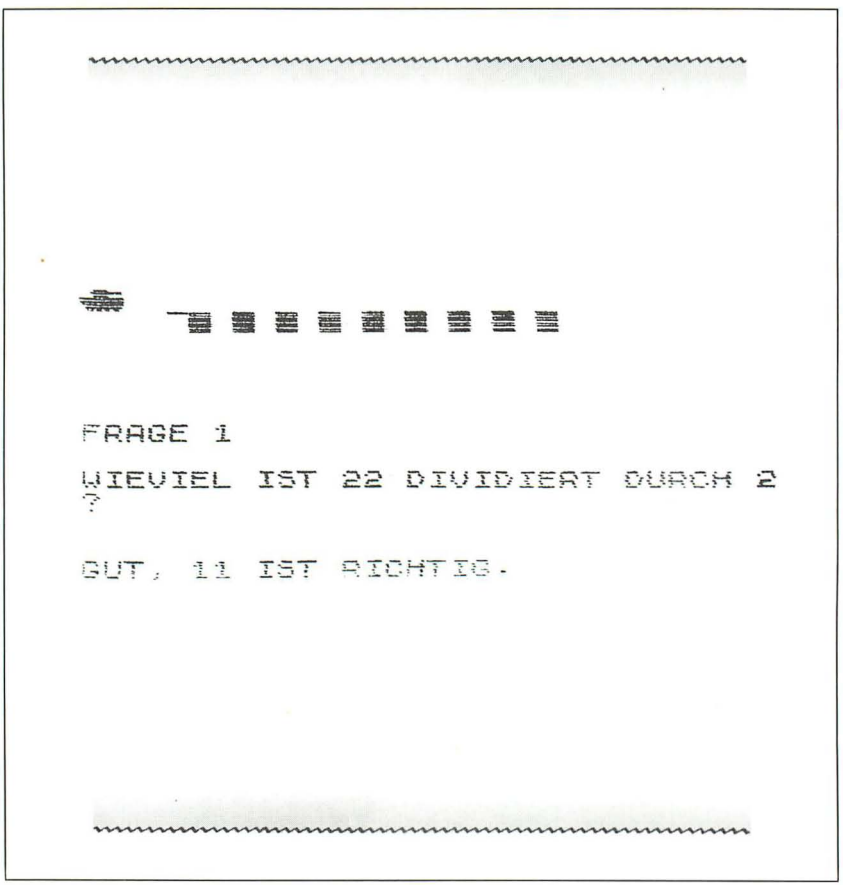
Das DIVISIONS-Programm wird jetzt um folgende Zeilen erweitert:

```
 140 IF s=20 THEN PAUSE 75: GO T
O 260
 145 STOP
 150 FOR n=1 TO 4: READ s$
 160 FOR m=0 TO 7: READ a: POKE
USR s$+m,a: NEXT m
 170 NEXT n
 180 DATA "a",7,15,7,255,255,127
,63,27
 190 DATA "b",192,254,192,255,25
5,255,254,108
 200 DATA "c",255,0,0,0,0,0,0,0
 210 DATA "d",0,16,0,254,0,16,0,
0
 220 LET b$="
 230 PRINT AT 3,0;"AB";: PRINT P
APER 4;"     ";: PRINT AT 4,5;"■ ■
 ■ ■ ■ ■ ■ ■ ■ ";: PRINT AT 4,24
;: PAPER 4;"          ";: PRINT PAPE
R 1;b$;b$
 240 GOTO 10
 250 PRINT AT 4,3+s; OVER 1;"C"
 RETURN
 260 FOR n=0 TO 10: FOR m=4 TO 8
 STEP .5: BEEP 0.01,m*m: NEXT m
 NEXT n
 270 FOR n=0 TO 23: PRINT AT 3,n
;" AB";: PAUSE 5: NEXT n
 280 FOR n=1 TO 50: PLOT 216,150
 : DRAW INK 2;36,0: PLOT 216,150
 DRAW INK 7;36,0: NEXT n
 290 PRINT AT 9,10;"GUT GEMACHT!
"
```

Die Zeilen 150–170 legen vier graphische Zeichen fest, die der Benutzer im einzelnen zuordnet. Die Daten dafür sind in den Zeilen 180 bis 210 enthalten. Mit b$ (in Zeile 210 definiert) löschen Sie in Zeile 110 lediglich einen Teil des Bildschirms. Das Bild von der Brücke und dem Panzer bleibt erhalten. Es wurde mit Zeile 230 „gemalt". Bevor sie AB tippen, müssen Sie mit dem Cursor den Panzer zeichnen. Das gleiche gilt für das C in Zeile 250 und für das AB in Zeile 270.

Zeile 250 zeichnet nach jeder richtigen Antwort ein weiteres Stück der Brücke. Zeile 260 spielt die Siegeshymne, wenn alle Antworten richtig sind. Zeile 270 läßt den Panzer über die Brücke fahren, und mit Zeile 280 feuert er.

Hier sehen Sie den Panzer in Aktion:



FRAGE 12

WIEVIEL IST 12 DIVIDIERT DURCH 4
?

FALSCH. DIE ANTWORT IST NICHT 2
(ES IST NAEMLICH 4 * 3 = 12)

SIE ERREICHTEN 20 PUNKTE VON
20 MOEGLICHEN!

GUT GEMACHT!

Sie können natürlich die Szene mit dem Panzer zur besseren Motivation der Schüler mit jedem Lernprogramm verbinden. Wenn Sie es aber zu oft benutzen, verliert es seinen Reiz. Versuchen Sie, selbst etwas zu erfinden, aber beachten Sie dabei ein paar wichtige Punkte. Alles, was zur Belohnung gehört, darf nicht die Lernelemente des Programms beeinträchtigen. Es darf auch nicht länger als ca. eine Sekunde nach jeder Frage dauern; sonst wird's langweilig. Auch sollte der Anreiz positiver Art sein: Es ist besser, wenn der Schüler durch Belohnungen ermutigt wird und weiter versucht, erfolgreich zu sein, als daß er für etwaiges Versagen bestraft wird, so daß er in ständiger Angst vor Mißerfolgen lernt. So macht Lernen nämlich keinen Spaß.

Legen Sie sich beim Schreiben von Erziehungsprogrammen nicht auf eine Möglichkeit fest. Der Computer muß z.B. nicht immer beides tun: Aufgaben stellen und Lösungen prüfen. Das nächste Programm GLEI-

```
   10 REM GLEICHUNGEN
   20 INK 7: PAPER 1
   30 BORDER 1: CLS
   40 DEF FN m(n)=INT (RND*n)+1
   50 LET a=FN m(10)
   60 LET b=FN m(10)
   70 LET c=FN m(10)
   80 LET d=FN m(10)
   90 LET x=FN m(10)
  100 LET y=FN m(10)
  110 LET e=a*x+b*y
  120 LET f=c*x+d*y
  130 PRINT '''' INK 6;"DIE GLEICH
UNGEN SIND:"
  140 PRINT 'TAB 8; INVERSE 1;a;"
x + ";b;"y = ";e
  150 PRINT 'TAB 8; INVERSE 1;c;"
x + ";d;"y = ";f
  160 PRINT '"DIESE MUESSEN NACH
"; INVERSE 1;"x"; INVERSE 0; UN
D "; INVERSE 1;"y"; INVERSE 0;"
AUFGELOEST WERDEN"
  170 PRINT ''"DRUECKEN SIE EINE
TASTE FUER DIE LOESUNG"
  180 PAUSE 0
  190 CLS
  200 PRINT '''''TAB 8;a;"x + ";b;
"y = ";e
  210 PRINT 'TAB 8;c;"x + ";d;"y
= ";f
  220 PRINT 'TAB 8;"x IST GLEICH
"; FLASH 1;x
  230 PRINT 'TAB 8;"y IST GLEICH
"; FLASH 1;y
  240 PRINT TAB 8;a;"*";x;" + ";b
;"*";y;" = ";e
  250 PRINT' TAB 8;c;"*";x;" + ";
d;"*";y;" = ";f
  260 PRINT ''''"DRUECKEN SIE BITT
E EINE TASTE   FUER EINE NEUE AU
FGABE"
  270 PAUSE 0
  280 RUN
```

CHUNGEN (für Gleichungen des Typs ax+by=z) stellt z.B. nur eine Aufgabe. Sie wird auf den Bildschirm geschrieben. Wenn der Schüler die Antwort sehen will, drückt er nur eine beliebige Taste (außer BREAK) und erhält die Lösung. Dabei sind die entsprechenden Werte in die Ausgangsgleichung eingesetzt. Sie könnten dieses Programm Ihrem Spectrum eingeben. Zur Übung können Sie durchgehend Dialog-Teile und einen Punktestandzähler sowie ein Hilfsmittel einbauen, um die Lösungen des Schülers entgegenzunehmen und auszuwerten.

Probestart: GLEICHUNGEN

```
 DIE GLEICHUNGEN SIND:

          3x + 1y = 13

          7x + 1y = 25

DIESE MUESSEN NACH x UND y AUFGE
LOEST WERDEN

DRUECKEN SIE EINE TASTE FUER DIE
LOESUNG


          3x + 1y = 13

          7x + 1y = 25

          x IST GLEICH 3
          y IST GLEICH 4

          3*3 + 1*4 = 13

          7*3 + 1*4 = 25

DRUECKEN SIE BITTE EINE TASTE
FUER EINE NEUE AUFGABE
```

Unser nächstes Programm KATZEN UND ANDERE DINGE ist für ganz kleine Kinder gedacht. Durchgehend wird INKEY$ benutzt. Die Input-Prüfung ist streng. Eine frei gewählte Anzahl Panzer, Lastwagen oder Katzen erscheint auf dem Bildschirm. Die richtige Anzahl muß eingetippt werden.

Beachten Sie den Unterschied beim Gebrauch von INPUT und INKEY$. Mit dem INPUT-Befehl können Sie Fehler während der Eingabe korrigieren. INKEY$ hat den Vorteil, daß Sie zur Eingabe von einzelnen Zeichen nicht ständig ENTER drücken müssen. Dadurch kann das Programm schneller abgearbeitet werden. Es sieht auch professioneller aus.

Die Panzer, Lastwagen oder Katzen werden mit den vom Benutzer erstellten graphischen Zeichen A und B geschrieben; die dafür nötigen Daten stellen die Zeilen 510 bis 530 bereit. Sie wählen die Art des Gegen-

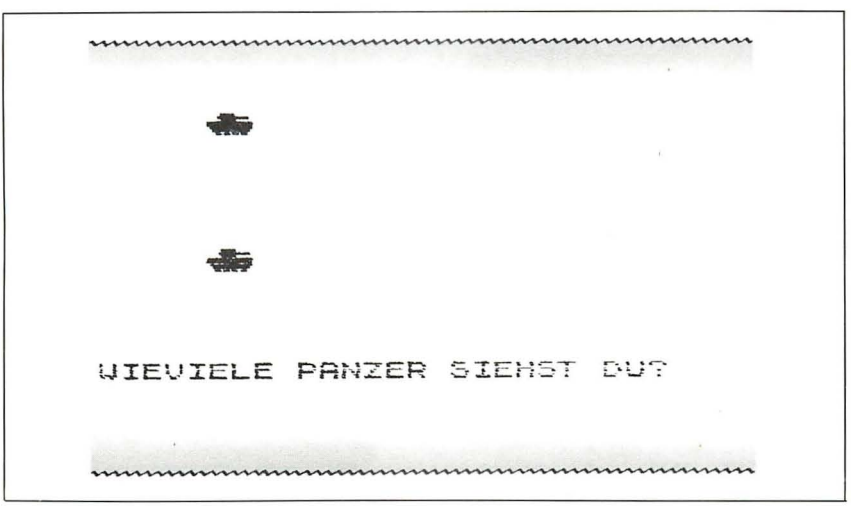
KATZEN UND ANDERE DINGE

```
  5 LET b$="
"
 10 LET i=1+INT (RND*3)
 15 REM AUSWAHL UND ZUSAMMENBAU
 EINES OBJEKTES
 20 RESTORE 500+10*i
 30 READ n$
 40 FOR n=1 to 2: READ s$
 50 FOR m=0 TO 7
 60 READ a: POKE USR s$+m,a
 70 NEXT m: NEXT n
 75 REM AUSGEBEN ZUFAELLIGE ZAH
L DER OBJEKTE
 80 LET z=1+INT (RND*9)
 90 RESTORE 600
100 FOR n=1 TO z
110 READ x,y
120 PRINT AT x,y,"AB"
130 NEXT n
140 PRINT AT 15,0,"WIE VIELE ";
n$;" SIEHST DU?"
150 GO SUB 410
160 IF CODE i$<48 OR CODE i$>57
THEN GO TO 150
170 IF i$=STR$ z THEN GO TO 250
175 REM KORREKTURTEIL
180 PRINT AT 15,0,b$ RESTORE 6
00
190 FOR n=1 TO z
200 READ x,y
210 PRINT AT x,y-1; PAPER 6,n
220 BEEP 1,n
230 NEXT n
240 GO TO 280
245 REM LOB
250 PRINT AT 15,0,"SEHR GUT.";b
$; PRINT AT 7,0; FLASH 1;z
260 FOR m=0 TO 4 STEP .1 BEEP
0.02,m*m: NEXT m: BEEP 1,20
270 PRINT AT 7,0;z
280 PRINT AT 21,0;"TIPPE BITTE
'ENTER' !"
290 GOSUB 410
300 IF CODE i$<>13 THEN GO TO 2
90
310 CLS : GO TO 10
400 REM EINGABE
410 IF INKEY$<>"" THEN GO TO 41
0
420 IF INKEY$="" THEN GO TO 420
430 LET i$=INKEY$
440 RETURN
500 REM DATEN
510 DATA "panzer","a",7,15,7,2
55,255,127,63,27,"b",192,254,192
,255,255,255,254,108
520 DATA "katzen","a",0,128,127
,63,63,56,40,40,"b",28,28,252,24
0,240,80,80,80
530 DATA "lkw's","a",0,0,0,255,
255,255,255,48,"b",28,30,30,255,
255,255,255,12
600 DATA 4,5,10,5,4,11,10,11,7,
8,4,18,10,18,4,24,10,24
```

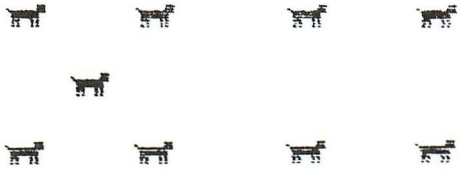
standes in Zeile 20 nach Belieben. Mit RESTORE ordnet der Computer dann den DATA-Zeiger der richtigen Zeile zu. Die Zeilen 30 bis 70 weisen dann n$ den Namen des Gegenstandes zu und legen die graphischen Zeichen fest. Die Zeilen 80–140 wählen die Anzahl der Objekte und Zeichen auf dem Bildschirm entsprechend den in Zeile 600 angegebenen Positionen. Dann wird das Kind gefragt, wie viele Objekte zu sehen sind. Beachten Sie, daß Sie für die Eingabe von AB in Zeile 120 den G-Cursor benötigen. BEEP wird nur in Verbindung mit dem sichtbaren Bild benutzt. Die Aufmerksamkeit des Kindes soll nämlich auf den sichtbaren Bereich gerichtet sein (Zeilen 180 bis 230). Mit ansteigender Tonfolge in den Zeilen 250 und 260 wird Lob ausgesprochen.

Sie könnten dieses Programm noch interessanter gestalten, wenn Sie Punkte zählen und nach jeweils 10 Fragen den Punktestand ausdrucken lassen. Sie können auch die Bildschirmdarstellung attraktiver machen, indem Sie z.B. in Zeile 140 eine INK-Farbe nach Ihrer Wahl einführen. Zeichnen Sie aber nicht weiße Katzen auf weißem Grund. Kinder fühlen sich betrogen, wenn sie dadurch beim Zählen verwirrt werden. Nehmen Sie noch andere Arten von Objekten dazu – wie wär's mit Vögeln oder Spinnen? Geben Sie ihnen kariertes Papier und lassen Sie sie etwas selbst ausdrucken. Schreiben Sie zusätzliche DATA-Zeilen ab 540. Geben Sie statt der Zahl 3 in Zeile 10 die Gesamtzahl der verschiedenen Objektarten ein. Und wenn Sie glauben, daß ein dominoartiges Muster zum Denken in Formen und Strukturen anregt, dann versuchen Sie noch eine andere Programmvariante: Lassen Sie die Objekte auf dem Bildschirm freiwählbare Positionen einnehmen.

Beispiellauf: KATZEN UND ANDERE DINGE



WIEVIELE LKW'S SIEHST DU?



WIEVIELE KATZEN SIEHST DU?

TIPPE BITTE 'ENTER

Bis jetzt haben wir uns in diesem Kapitel auf numerische Aufgaben konzentriert. Die sind am einfachsten zu schreiben. Es ist aber auch nicht allzu schwierig, Programme mit verbalen Antworten für den Spectrum zu schreiben.

Das nächste Programm FRANZÖSISCH-VOKABULAR speichert Fragen und Antworten in DATA-Befehlen. Es kann leicht ergänzt oder auf ähnliche Frage- und Antwortspiele übertragen werden. Denken Sie daran, daß in diesem Programm CAPS LOCK enthalten sein muß. Denn die vom Computer überprüften Fragen und Antworten werden alle in Großbuchstaben geschrieben. In Zeile 80 des Programms kann der Schüler die Fragen wählen, an die er sich heranwagen will. Diese Zeile brauchen Sie für die FOR/NEXT-Schleife in Zeile 100.

Sie glauben vielleicht, es sei wichtig für Programme dieser Art, sicherzustellen, daß die gleiche Frage nicht mehr als einmal pro Durchlauf gestellt wird. Vielleicht wollen Sie dann den in diesem Programm angewandten Mechanismus, der dies gewährleistet, benutzen oder anpassen. Zeile 30 stellt eine Reihe auf und setzt, wie Sie wissen, jedes Glied der Reihe gleich Null. In den DATA-Befehlen sehen Sie, daß auf jedes deutsche Wort ein französisches, und diesem jeweils eine Zahl folgt. Bei jeder Wahl eines Wortpaares (indem Sie zufällig den Data-Befehl durchlesen, siehe Zeile 110 bis 130) wird der Variablen x ein Wert zugeordnet. Zu jedem Wortpaar gehört eine Zahl. Diesem Element wird in Zeile 160 der Wert 1 zugeordnet. Nachdem Zeile 150 kontrolliert hat, ob das schon passiert ist, wird die Frage ausgedruckt und mit dem Befehl RESTORE wieder neu gestellt. Ein neues Wortpaar wird gewählt (einschließlich dazugehöriger Nummer). Am Ende eines Quiz ist es zwar etwas schwerer als am Anfang, ein noch nicht gebrauchtes Wortpaar zu finden, aber sie werden beim Durchlauf schon sehen, daß die dadurch verursachte Verzögerung das Programm eher bereichert, als ihm abträglich ist.

Auch dieses Programm entfaltet bei einer richtigen Antwort ein Feuerwerk. Beachten Sie, daß eine falsche Antwort des Schülers berichtigt wird.

```
  10 REM FRANZOESISCHE VOKABELN
  20 LET punkte=0: PAPER 1: INK
7: BORDER 1: CLS
  30 DIM a(20)
  40 PRINT '''TAB 6;"WILLKOMMEN B
EIM "
  50 PRINT TAB 6;"VOKABELTEST!"
  60 PRINT '''TAB 6;"WIEVIELE VOK
ABELN"
  70 PRINT TAB 6;"MOECHTEN SIE T
ESTEN (1-20)"
  80 INPUT b
  90 IF b<1 OR b>20 THEN GO TO 8
0
```

```
 100 FOR c=1 TO b
 110 FOR d=1 TO INT (RND*20)+1
 120 READ e$: READ f$: READ x
 130 NEXT d
 140 RESTORE: CLS
 150 IF a(x)=1 THEN GO TO 110
 160 LET a(x)=1
 170 PRINT ''' INK RND*7; PAPER
9;" FRAGE NR. ";c;": "
 180 PRINT '''"WIE HEISST AUF FR
ANZOESISCH";TAB 4;e$;"?"
 190 INPUT a$
 200 IF a$=f$ THEN LET punkte=pu
nkte+1: BEEP .1,2.5*punkte: PRIN
T FLASH 1; INK 2; PAPER 5; ''"JA
,";f$;" IST RICHTIG "
 210 IF a$<>f$ THEN PRINT '''"NEI
N, LEIDER FALSCH."''"DAS FRANZOE
SISCHE WORT FUER      ";e$;" IST
";f$
 220 IF c<b THEN PRINT '' INK 3;
 FLASH 1; BRIGHT 1;" IHR PUNKTES
TAND IST NUN ";punkte;" "
 230 PAUSE 150
 240 NEXT c
 250 INK 2: PAPER 7: FLASH 1: CL
S : PRINT AT 6,3;"IN DIESEM TEST
 HATTEN SIE ";punkte,"     PUNKTE
"
 260 PAUSE 200: FLASH 0
 270 RUN
 280 DATA "UEBERALL","PARTOUT",1
,"SELTEN","RAREMENT",2,"OFT","SO
UVENT",3,"NIEMALS","JAMAIS",4,"I
MMER","TOUJOURS",5
 290 DATA "SEHR","TRES",6,"MEHR"
,"PLUS",7,"WENIGER","MOINS",8,"J
A","OUI",9,"NEIN","NON",10
 300 DATA "DANKE","MERCI",11,"EI
NS","UN",12,"ZWEI","DEUX",13,"DR
EI","TROIS",14,"VIER","QUATRE",1
5
 310 DATA "FUENF","CINQ",16,"SEC
HS","SIX",17,"SIEBEN","SEPT",18,
"ACHT","HUIT",19,"NEUN","NEUF",2
0
 320 STOP
```

Beispiellauf: FRANZÖSISCHE VOKABELN

```
WILLKOMMEN BEIM
VOKABELTEST.

WIEVIEL VOKABELN
MOECHTEN SIE TESTEN (1-20)
```

```
 FRAGE NR. 1

WIE HEISST AUF FRANZOESISCH
    MEHR?
PLUS

 JA, PLUS IST RICHTIG

 IHR PUNKTESTAND IST NUN 1



 FRAGE NR. 2

WIE HEISST AUF FRANZOESISCH
    UEBERALL?
PARTUOT

NEIN, LEIDER NICHT.

DAS FRANZOESISCHE WORT FUER
UEBERALL IST PARTOUT

IHR PUNKTESTAND IST NUN 1



 FRAGE NR. 3

WIE HEISST AUF FRANZOESISCH
    FUENF?
CINQ

JA, CINQ IST RICHTIG

IHR PUNKTESTAND IST NUN 2
```

Das nächste Programm QUIZMASTER können Sie für alles benutzen, von „Buchstabier"-Listen bis zur Hauptschulbiologie. Der Haken ist, daß Sie selbst die Fragen dazu liefern müssen. Das ist leichter, als Sie glauben.

Das Programm wird dazu benutzt, um eine Auswahl Fragen oder entsprechende Definitionen zusammen mit den richtigen Antworten zu speichern. Hier sind ein paar Beispiele, mit denen Sie anfangen können:

Wie heißt das französische Wort für „geben"?
(Antwort: donner)

Wie ist der Name des Minerals in St. Austin?
(Antwort: weißer Ton)

Welcher Atomkernteil hat die Masse 1 und keine Ladung?
(Antwort: das Neutron)

Welche Zellorganismen enthalten die Zellatmungsfermente?
(Antwort: Mitochondrien)

Für den Buchstabiertest dürfte es reichen, wenn die entsprechenden Definitionen, die einen Anhaltspunkt für das zu ratende Wort geben sollen, einfach sind: Die Angabe „bewaffneter Reiter" gibt beispielsweise einen recht guten Fingerzeig für den Begriff „Ritter" auf der Liste. Überreden Sie Ihr Kind dazu, eigene Anhaltspunkte und Wörter einzugeben. Dann gehen Sie die Übung selbst durch, um sicherzustellen, daß die Rechtschreibung stimmt.

Das Programm wirft mit FLASH zuerst die gesamte Menge der Antworten, eine nach der anderen, kurz auf den Bildschirm. Dann werden die Fragen in zufälliger Abfolge, aber diesmal ohne Antworten, vorgelegt. Wenn Sie einen Fehler machen oder einfach ENTER drücken, wird Ihnen ein Anhaltspunkt gegeben: „R....r" für Ritter oder „d....r" für donner (geben). Mit Run 500 können Sie eine neue Frage- und Antwort-Serie aufstellen.

QUIZ-MASTER

```
  10 REM INITIALISIERUNG
  20 LET s=0: LET x=m: LET b$="-
--------------": LET e$="
                              "
  30 FOR n=1 TO m: LET y(n)=n: N
EXT n
  40 GO SUB 400: CLS : FOR n=1 T
O m: PRINT AT 10,8;a$(n): PAUSE
50: CLS : NEXT n
  50 REM HAUPTPROGRAMM
  60 LET z=INT (RND*x+1): LET q=
y(z)
  70 LET x=x-1
  80 FOR n=z TO x: LET y(n)=y(n+
1): NEXT n
  90 CLS : LET t=2
 100 PRINT AT 3,0;c$(q): INPUT i
$
 110 IF i$<>a$(q)( TO t(q)) THEN
```

```
 GO TO 240
 120 LET s=s+t: GO TO 210
 130 GO SUB 410: CLS
 140 IF x>0 THEN GO TO 60
 150 REM ENDSTAND
 160 PRINT AT 5,0;"SIE HABEN ";s
;" VON ";m*2;" PUNKTEN"
 170 STOP
 200 REM LOB
 210 PRINT AT 7,0;"GUT GEMACHT,
";i$;" WAR RICHTIG.";AT 9,15;e$
 220 GO TO 130
 230 REM FALSCHE ANTWORT
 240 IF i$="" THEN GO TO 260
 245 PRINT AT 7,0;i$;" WAR FALSC
H.";e$;AT 9,0;e$
 250 GO SUB 410
 260 IF t=2 THEN GO TO 310
 270 PRINT AT 7,0;"DIE RICHTIGE
ANTWORT WAR:";e$;AT 9,15;a$(q)
 280 GO TO 130
 300 REM HINWEIS/ZWEITER VERSUCH
 310 LET t=1
 320 PRINT AT 7,0;"HIER EIN HINW
EIS:";e$;AT 21,0;e$
 330 PRINT AT 9,15;a$(q,1)+b$(1
TO l(q)-2)+a$(q,l(q))
 340 GO TO 100
 400 REM FORTSETZUNG
 410 PRINT AT 21,0;"ZUR FORTSETZ
UNG ENTER EINGEBEN!"
 420 GO SUB 460
 430 IF CODE k$=13 THEN RETURN
 440 GO TO 420
 450 REM EINE TASTE EINGEBEN
 460 IF INKEY$<>"" THEN GO TO 46
0
 470 IF INKEY$="" THEN GO TO 470
 480 LET k$=INKEY$: RETURN
 500 REM EINGABE DER FRAGEN UND
ANTWORTEN
 510 PRINT "WIEVIELE FRAGEN?": I
NPUT m
 520 DIM c$(m,64): DIM a$(m,15):
 DIM l(m): DIM y(m)
 530 FOR q=1 TO m
 540 CLS : PRINT "FRAGE ";q''
 550 PRINT "GEBEN SIE BITTE DIE
FRAGE ODER  UMSCHREIBUNG AN!": I
NPUT c$(q)
 560 PRINT'c$(q)''"GEBEN SIE BIT
TE DIE ANTWORT EIN!": INPUT I$
 570 PRINT 'i$;AT 20,0;"TIPPEN S
IE  O, WENN DIE FRAGE OKAY IST,
ODER N, WENN SIE DIE EINGABE NOC
HMAL MACHEN MOECHTEN!"
 580 GO SUB 460
 590 IF k$="N" THEN GO TO 540
 600 IF k$<>"O" THEN GO TO 580
 610 LET l(q)= LEN i$: LET a$(q)
=i$
 620 NEXT q
 650 REM SPEICHERN UND STARTEN
 660 CLS : INPUT "NAME DES PROGR
AMMS?";p$
 670 SAVE p$ LINE 20
```

Um den Ablauf des Programms gut zu verstehen, sollten Sie am besten bei Zeile 500 anfangen, wo die Fragen und Antworten eingegeben werden. Die Gesamtzahl der Fragen m wird in Zeile 510 gegeben. Sie wird dazu benötigt, um vier Felder in Zeile 520 anzulegen. Feld c$ enthält m Fragen, jede bis zu 64 Zeichen (2 Zeilen) lang. Feld a$ enthält m Antworten, jede 15 Zeichen lang. Für längere Fragen oder Antworten müssen Sie den DIM-Befehl ändern. Feld 1 gibt die reale Länge der Antworten an: Die Antwort-Zeichenketten werden mit Leerstellen ausgestattet, wenn sie in dem Feld gespeichert werden. Die Antwort-Länge ist nötig, um die Antwort des Kindes mit der richtigen Lösung vergleichen zu können. Die Lösungen werden eingangs mit Input i$ akzeptiert. So kann ihre Länge vor ihrer Speicherung in Feld a$ bestimmt werden.

Die Zeilen 660 und 670 zwingen Sie, Ihr Programm zu speichern. Die Fragen und Antworten können dann nur noch mit RUN gelöscht werden. Wenn Sie das Programm von neuem starten wollen, müssen Sie den Befehl GO TO 20 geben, denn dann bleiben die Fragen und Antworten erhalten. Mit ein paar zusätzlichen Zeilen am Ende des Hauptprogramms können Sie das Programm jederzeit wieder starten.

Jetzt gehen wir wieder zum Ausgangspunkt des Programms zurück. Die Zeilen 20 und 30 stellen Variablen vor: s für die Punktzahl, x für die Anzahl der Fragen, die noch gestellt werden müssen, b$, um die Anzahl der Bindestriche in den Hinweisen festzulegen, und e$ zum Löschen einzelner Zeilen auf dem Bildschirm.

Zeile 40 schreibt alle Antworten mit FLASH auf den Bildschirm. Das ist eine gute Gedächtnishilfe für kurze Frage-Serien. Machen Sie das aber nicht, wenn es allzu viele Fragen sind.

Das Feld Y (in Zeile 520) speichert die Fragenummern so, daß ärgerliche Wiederholungen beim Programmdurchlauf vermieden werden. Zeile 30 legt die Elemente wie folgt fest: y(1) = 1, y(2) = 2 usw. Bei nur vier Fragen in der Übung gibt Zeile 60 dem z einen Wert zwischen 1 und 4. Angenommen, daß z = 2 ist, dann würde die Fragenummer q mit y(2) gleichgesetzt und erhielte so den Wert 2. Zeile 70 läßt x kleiner werden, d. h. die Zahl der noch verbleibenden Fragen. Zeile 80 löscht die Nummern der Fragen, die schon verwendet wurden:

vor Zeile 80	nach Zeile 80
y(1) = 1	y(1) = 1
y(2) = 2	y(2) = 3
y(3) = 3	y(3) = 4
y(4) = 4	y(4) = 4

Sie bekommen zwei Punkte, wenn Sie die Frage auf Anhieb, und einen Punkt, wenn Sie sie beim zweiten Anlauf richtig beantworten. Zeile 110

benutzt die Länge l(q) der Antwort, um die Leerstellen aus der Antwort zu entfernen. Sie müssen diese Leerstellen entfernen, weil

„Constantinople“ <> „Constantinople “ ist.

Sie brauchen die Länge der Antwort in Zeile 330 noch einmal, um für die Vorgabe „C---------“ die richtige Anzahl Bindestriche von b$ vorzulegen.

Dieses Programm kann dazu benutzt werden, um den Schülern in vielen Fächern bei den Schulaufgaben zu helfen. Wenn Sie Ihren Spectrum regelmäßig mit Französisch-Vokabeln, Chemie-Definitionen und Fragen zur englischen Literatur füttern, dann haben Sie sich bald einen beträchtlichen Grundstock an Wiederholungsmaterial aufgebaut (dabei werden Sie selbst wahrscheinlich auch ein wenig lernen). Das einleitende FLASH bei den Antworten und die Hinweise führen dazu, daß Sie sich bei Wiederholungen des Programms im Laufe des Jahres gut erinnern werden. Das gibt dem Anwender Sicherheit beim Gebrauch grundlegender Begriffe und Strukturen, ohne die er in keinem Fach Fortschritte machen kann. Um das Programm noch zu erweitern, könnten Sie in einem zusätzlichen Feld Erläuterungen und ergänzende Informationen speichern. Der Benutzer kann sie dann ganz nach Belieben abrufen oder unberücksichtigt lassen. Man könnte diese Programme noch auf eine andere Art und Weise weiterentwickeln: Man prüft eine Reihe falscher Antworten anhand einer Liste verwandter Begriffe nach. Der Spectrum soll dann die richtige Bedeutung für die Antwort herausfiltern.

Von diesem sehr allgemein gehaltenen Lehrprogramm gehen wir jetzt über zu einem ganz anderen, sehr speziellen. Es testet die Lesegeschwindigkeit und die Aufnahmefähigkeit für das Gelesene. Die Idee stammt aus dem Buch „The Standard Reading Tests“. Der Computer trifft seine Wahl unter Sätzen, die in DATA-Befehlen ab Zeile 160 enthalten sind, und bringt sie auf den Bildschirm. Dort bleiben sie entsprechend ihrer Länge (siehe Zeile 60) eine Zeitlang stehen. Dann wird der Bildschirm gelöscht. Der Schüler wird dazu aufgefordert, den Satz einzugeben, den er gerade auf dem Bildschirm gesehen hat. Der vom Schüler eingegebene Satz (der Anfangsbuchstabe kann groß oder klein geschrieben werden) wird mit dem Satz des Computers verglichen (mit Ausnahme des Anfangsbuchstaben).

Wenn beide übereinstimmen, meldet der Bildschirm „Gut gemacht“. Wenn ein Satz gelöscht ist, kommt der nächste an die Reihe. Hat der Schüler etwas falsch gemacht, dann wird der Satz nach einer kurzen Pause wiederholt. Das geschieht so oft, bis der Satz richtig eingegeben ist.

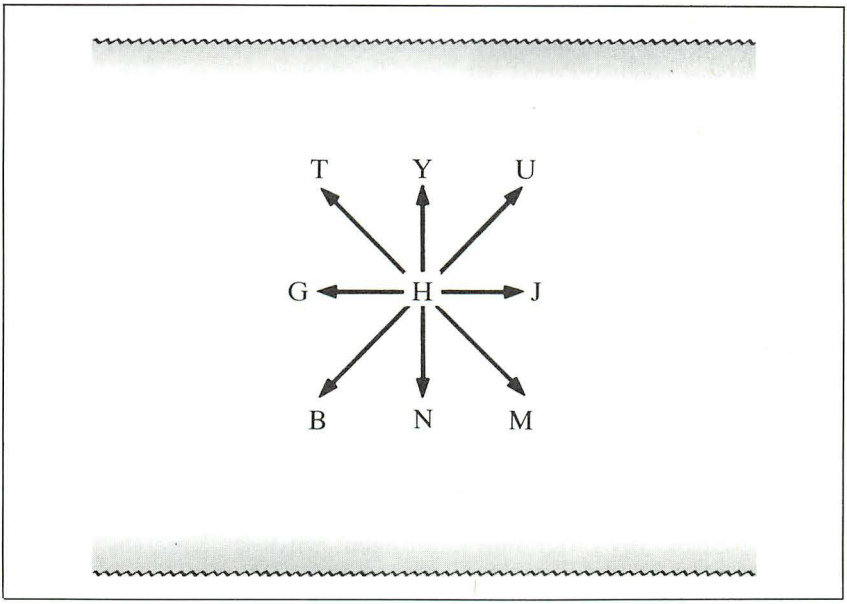
Sie können die 20 am Ende der Zeile 60 ändern, um den Satz so lange auf dem Bildschirm stehenzulassen, wie es für die Lesefähigkeit des Schülers nötig ist. Auch die Sätze selbst sollten jeweils an das Alter und die Fähigkeiten des Schülers angepaßt werden. Sie können sich auch stellvertre-

tend für den Computer die Sätze mündlich wiederholen lassen. Dann brauchen die Sätze nicht eingetippt zu werden, wenn das Ihrer Meinung nach eine bessere Übung ist.

LESETEST

```
  10 INK 1: PAPER 7: BORDER 7: C
LS
  20 PAUSE 50: RESTORE
  30 FOR j=1 TO INT (RND*11)+1
  40 BEEP 1/j,3*j: READ a$: NEXT
 j
  50 PRINT AT 10,0;a$
  60 PAUSE (LEN a$)*20
  70 CLS
  80 INPUT INK 2;"GEBEN SIE JETZ
T DEN TEXT EIN,   DEN SIE GERADE
 GELESEN HABEN",,b$
  90 PRINT INVERSE 1;" ";b$;" "
 100 PAUSE 50
 110 IF a$(2 TO )=b$(2 TO ) THEN
 BEEP .3,LEN a$: PRINT '' INK 3;
FLASH 1;"GUT GEMACHT, DAS WAR RI
CHTIG!": PAUSE 300: RUN
 120 PRINT ''"BEDAURE, DAS WAR N
ICHT RICHTIG!"
 130 PRINT ''"HIER IST ES NOCHMA
L"
 140 PAUSE 200: CLS
 150 GO TO 50
 160 DATA "VORSICHT BISSIGER HUN
D"
 170 DATA "DIE KATZE SCHLEICHT U
M DEN BREI"
 180 DATA "WARUM IST DAS EI BRAU
N?"
 190 DATA "DIE SONNE BRENNT HEIS
S"
 200 DATA "FISCHERS FRITZE FISCH
T FISCHE"
 210 DATA "DIE ENTE SCHWIMMT SCH
NELL"
 220 DATA "DAS KAMEL HAT EINEN H
OECKER"
 230 DATA "ES IST GLEICH 5 VOR 1
2"
 240 DATA "NICHT NUR FISCHE HABE
N SCHUPPEN"
 250 DATA "DER KLAVIERSTIMMER KL
IMPERT"
 260 DATA "DER BUSFAHRER BUMMELT
"
```

Mit dem nächsten Programm DUDELSACK machen Kinder gern Skizzen und Bilder. Wir zeichnen jetzt damit Landkarten und Diagramme. Mit den unten angegebenen Tasten können Sie waagerechte, senkrechte und diagonale Geraden zeichnen:



Die Taste H im Zentrum hat keine Bedeutung.

DUDELSACK

```
  10 BORDER 6
  20 INPUT "ANFANGSKOORDINATE X:
";x
  30 INPUT "ANFANGSKOORDINATE Y:
";y
  40 PLOT x,y
  70 LET c$=INKEY$
  80 LET x=x+(c$="u")+(c$="j")+(
c$="m")-(c$="t")-(c$="g")-(c$="b
")
  90 LET y=y+(c$="t")+(c$="y")+(
c$="u")-(c$="b")-(c$="n")-(c$="m
")
 100 LET x=x-(x>255)+(x<0)
 110 LET y=y-(y>175)+(y<0)
 120 IF c$="s" THEN STOP
 130 GO TO 40
```

Geben sie das Programm ein und starten Sie. Wählen Sie Ihre Startposition mit der x- und y-Punktkoordinate. Zeile 70 kontrolliert, welche Taste Sie drücken, und ordnet ihr c$ zu. Entsprechend ändern sich die x- und y-Koordinaten in Zeile 80 und 90. Wenn Sie „m“ drücken. dann ist (c$ =

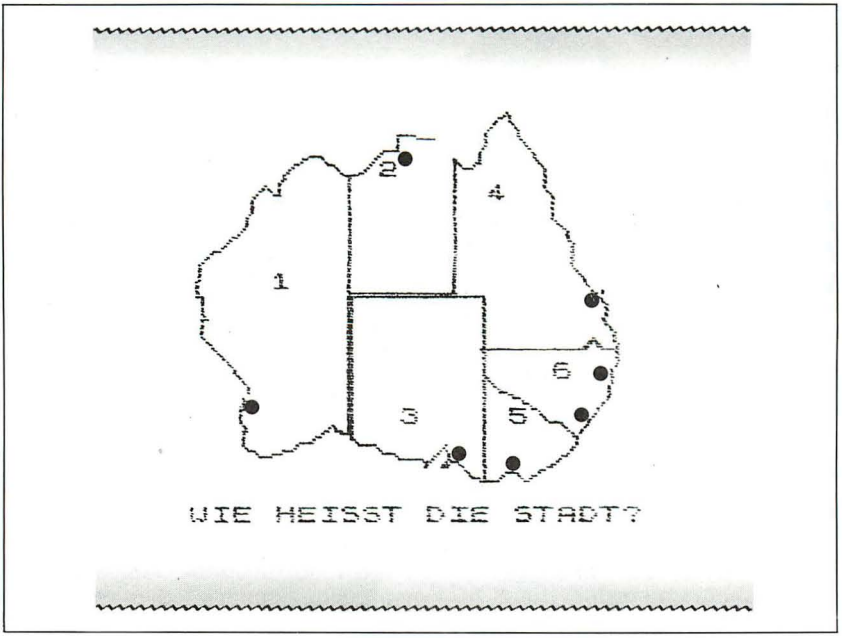
„m“) wahr und nimmt den Wert 1 an. Der restliche Inhalt der Zeilen 80 und 90 ist nicht wahr und also gleich 0. Demnach sind x = x+1 und y = y+1, und der nächste Punkt wird eine Position weiter südöstlich gezeigt als der erste. Die Zeilen 100 und 110 verhindern, daß Sie über die Grenzen der Zeichenfläche hinausgehen. Dann würde nämlich alles „zusammenbrechen“. In Zeile 120 können Sie mit der Taste s das Skizzieren abbrechen.

Sie können Ihre Kunstwerke mit dem Befehl SAVE „Skizze“ SCREEN$ speichern. Unter der Überschrift „Skizze“ wird Ihr Bild jetzt auf Band gespeichert. Dieses simple „Skizzenblock“-Programm können Sie u. a. wie folgt verbessern. Schreiben Sie die Zeile 120 neu:

```
120  IF c$ = "c" THEN GO TO 140
```

Bei Taste „c“ geht das Programm zur Zeile 140. Dort können Sie z. B. Befehle wie INPUT „Ink-Farbe“; Tinte eingeben. Schreiben Sie in Zeile 40 nach dem PLOT-Befehl INK;. Jetzt können Sie in Farbe zeichnen. Wenn Sie zusätzlich ähnliche Befehle wie PAPER, OVER, FLASH, BRIGHT und INVERSE ins Programm einbringen, wird das Ergebnis entsprechend interessant ausfallen. Das zeigt folgendes Beispiel:

Probedurchlauf DUDELSACK:



Für die Karte von Australien wurde das ursprüngliche Skizzenblockprogramm abgewandelt. Zwei Zeilen sind neu hinzugekommen:

```
50  IF INKEY$ <> "" THEN GO TO 50
60  IF INKEY$ = "" THEN GO TO 60
```

Das verlangsamt das Programm, so daß mit jedem Tastendruck nur ein Punkt gezeichnet wird. Sie brauchen eine Landkarte, die genauso groß ist wie der Teil der Bildschirmfläche, auf dem gezeichnet wird. Übertragen Sie die Landkarte auf ein Stück Zeichenfolie und heften Sie sie auf Ihren Bildschirm. Etwas Platz muß für Fragen und Antworten frei bleiben.

Speichern Sie die fertige Skizze. Zur Vervollständigung Ihrer Landkarte können Sie direkte Befehle geben wie z. B.: CIRCLE 55,50,2. Das ergibt einen kleinen Kreis für eine Stadt. Mit Hilfe des Programms suchen Sie sich die Koordinaten für Orte auf der Landkarte. Ziehen Sie eine Linie zu dem gewünschten Punkt, halten Sie das Programm an und schreiben Sie die x- und y-Werte in eine Ecke des Bildschirms. Starten Sie das Programm wieder mit GO TO 40 und wiederholen Sie diesen Vorgang. Ge-

ben Sie zum Schluß noch einmal eine saubere Kopie der Landkarte ein, fügen Sie kleinere und größere Städte ein und speichern Sie diese Endversion.

Die gleiche Technik können Sie in fast jedem Fach anwenden, von Schaltungsdiagrammen in der Physik bis zu Querschnitten von Pflanzenstielen in Biologie.

Als nächstes sollen Sie Ihre Landkarte oder Ihr Diagramm mit einem Frage- und Antwort-Programm verbinden. Wenn Sie vom Quizmaster-Programm ausgehen wollen, müssen Sie es etwas abändern. Ersetzen Sie Zeile 40 durch:

```
40  LOAD "Landkarte" SCREEN$
```

Alle PRINT-Befehle müssen an das untere Ende des Bildschirms gepackt werden. Das läßt sich mit folgendem Unterprogramm bewerkstelligen:

```
350  REM 3 Zeilen am unteren Rand löschen
360  PRINT AT 19,0; e$; e$; e$
370  PRINT AT 19,0;
380  RETURN
```

Geben Sie mit RUN 500 Ihre Fragen und Antworten ein. Wenn Sie alles gespeichert haben, geben Sie den Befehl BREAK und holen Sie Ihre Landkarte wieder von der Kassette zurück auf den Bildschirm. Speichern Sie dies mit dem Befehl SAVE „Karte" SCREEN$ auf dieselbe Kassette direkt hinter dem Frage- und Antwort-Programm. Wenn sie jetzt das Programm QUIZMASTER laden, wird automatisch auch das Bild geladen. Danach geht das Programm sofort zu den Fragen über.

In diesem Kapitel haben wir also einige Programme entwickelt, die den Schülern sicherlich großen Spaß machen werden und die auch ohne großen Zeitaufwand eingetippt werden können. Diese Programme gehören zur Kategorie der Lern- und Übungsprogramme.

Um mit Erfolg zu lernen, braucht ein Schüler

1. viele Übungsmöglichkeiten
2. die sofortige Information, ob die Antwort richtig oder falsch ist
3. die Möglichkeit, das Lerntempo selbst festzulegen
   und, was sehr wichtig ist,
4. die Befriedigung, die sich bei erfolgreichen Ergebnissen einstellt.

Es sind auch andere Lehr- und Lernprogramme möglich, wie z. B. Programme, bei denen unter mehreren Antworten ausgewählt werden muß, oder solchen, bei denen Figuren zur darstellenden Geometrie verwendet

werden. Aber darauf wollen wir hier nicht eingehen. Das vorgelegte Material dürfte sicherlich eine gute Starthilfe zur Entwicklung eigener Software geben.

Wenn Ihnen die nötigen Ideen fehlen, bestellen Sie sich einen Katalog über Lernprogramme für irgendeinen Computer. Beim bloßen Durchlesen der Programmbeschreibungen wird Ihnen sicherlich die eine oder andere Idee kommen. Auch die vorliegenden Programme basieren zum Teil auf solchen Katalogen.

Nun sind wir am Ende unseres Kapitels angelangt. Wir haben einige einfache und doch recht wirkungsvolle Programme kennengelernt. Diese Programme können als Grundstock zur Entwicklung einer eigenen Programmbibliothek dienen. Sei es, daß Sie die Programme selber schreiben, oder sei es, daß Sie Programme kaufen. Sie sollten auf jeden Fall die folgenden Punkte beachten:

1. Sie sollten sich beim Programmstart nicht erst durch mehrere Erläuterungstafeln quälen müssen.
2. Der Bildschirm sollte nicht mit Informationen überladen sein. Nur die wesentlichen Informationen gehören auf den Bildschirm.
3. Beim Text sollten Farben nur dann verwendet werden, wenn dies sinnvoll erscheint.
4. Das Programm sollte abbruchsicher sein.
5. Wenn eine Antwort falsch ist, muß die richtige Antwort gezeigt werden.
6. Das Tempo der Bearbeitung sollte vom Anwender und nicht vom Programm bestimmt werden. So sollten Informationen erst dann vom Bildschirm verschwinden, wenn der Anwender sie gelesen hat. Außerdem sollten die Programme keine allzu langen Pausen zwischen den einzelnen Fragen machen.

Ein gutes Programm erkennt man daran, daß es die gesteckten Ziele bewältigt und dabei noch Freude macht.

Vorschläge zur weiteren Lektüre:

Orwig, Gary W. und Hodges, William S.: The Computer Tutor, Winthrop Publishers, Inc., USA, 1981

Daniels, J. C. und Diak, Hunter: The Standard Reading Tests, Chatto Educational Ltd., 1958

Rogowski, Stephan J.: Problems for Computer Solution, Creative Computing Press, USA, 1979

Maddison, Alan: Microcomputers in the Classroom, Hodder and Stoughton, London, 1982

Oliva, Ralph A. (Hrsg.): Understanding Calculator Math., Texas Instruments Inc., Dallas, Texas; vertrieben durch Tandy/Radio Shack, 1978

Kapitel 7

Der Spectrum als Spielpartner

Ganz gleich, aus welchen Gründen Sie Ihren Spectrum gekauft haben: Sie werden sicher auch gelegentlich ein Spiel mit ihm spielen wollen. Es gibt für einen Computerbesitzer kaum etwas Faszinierenderes, als sich eigene Programme (Spiele oder anderes) auszudenken und sie Schritt für Schritt in die Tat umzusetzen.

In diesem Kapitel wollen wir uns einige Spiele anschauen. Alle werden im Detail erklärt, damit Sie ein paar Tricks lernen, wie solche Programme geschrieben werden. Auf diese Weise werden Sie bald Ihre eigenen Spiel-Programme schreiben können.

Es ist zwar verständlich, wenn Sie gleich die fertige Version der Spielprogramme eingeben wollen, ohne vorher alle Erläuterungen davor und dahinter durchzulesen. Wenn Sie das machen, werden Sie zwar ein lauffähiges Programm erhalten, aber das ist nicht der Sinn dieses Kapitels. Haben Sie lieber ein wenig Geduld. Folgen Sie Zeile für Zeile der Beschreibung und geben Sie bei den schrittweise erläuterten Programmen jeden Schritt erst dann ein, wenn Sie zu der entsprechenden Anleitung kommen.

Die ersten beiden Programme NÄCHTLICHER ÜBERFALL und REBENKLAU sind besonders ausführlich erklärt. Diese beiden Beschreibungen sollten Sie auf jeden Fall sehr sorgfältig lesen, selbst wenn Sie die Erläuterungen der nachfolgenden überspringen. Die beiden Programme enthalten sehr viele nützliche Ideen, die nicht nur zum besseren Verständnis der Programme beitragen, sondern auch bei anderen Programmierungs-Problemen Anwendung finden.

Nächtlicher Überfall

Im ersten Spiel fliegt ein Flugzeug über eine Stadt mit dem Ziel, die Wolkenkratzer dem Erdboden gleich zu machen. Das Flugzeug fliegt über den Bildschirm und verliert bei jedem Zeilenende an Höhe. Schließlich

stürzt es beim Zusammenprall mit einem Wolkenkratzer ab, falls Sie nicht schon vorher das Gebäude zerstört haben.

Zunächst einmal müssen wir die Wolkenkratzer zeichnen. Das geht sehr einfach mit einer FOR/NEXT-Schleife von 0 bis 31 (dies entspricht den Spalten des Bildschirms). Diese Schleife muß eine weitere Schleife enthalten, mit der die Wolkenkratzer von unten nach oben aufgebaut werden.

Zur Verdeutlichung möge folgendes Programm dienen:

```
10 FOR a=0 TO 31
20 FOR b=11 TO 21
30 PRINT AT b,a;"■"
40 NEXT b
50 NEXT a
```

Die Wolkenkratzer, die Sie jetzt sehen, sind alle gleich hoch. Sie können die Höhe des gesamten Häuserblocks variieren, indem Sie die Werte von b in der Zeile 20 entsprechend ändern.

Mit dieser Routine haben Sie noch immer keine interessante Silhouette. Das Ganze sieht eher wie ein Schuhkarton und nicht wie eine Stadt aus. Um den Wolkenkratzern unterschiedliche Höhen zu geben, brauchen wir einen Zufallsfaktor. Versuchen Sie einmal, folgendes in Zeile 20 einzugeben:

```
20 FOR b = INT (RND*22) TO 21
```

Die Silhouette bekommt jetzt schon Konturen. Das sieht schon besser aus, wirkt aber zu unregelmäßig. Was wir brauchen, ist ein Durchschnittswert, eine mittlere Höhe. Der Zufallsfaktor soll dann diese Höhe geringfügig variieren. Jetzt entscheidet nicht mehr der Zufallsfaktor über den Spielverlauf, sondern wir legen durch die mittlere Höhe den Schwierigkeitsgrad des Spiels fest. Um eine Stadt zeichnen zu können, benötigen wir also die mittlere Bebauungshöhe. Einige der Wolkenkratzer werden etwas höher ausfallen, andere werden geringfügig unter dieser Richthöhe liegen.

Im letzten Beispiel lag die mittlere Höhe bei elf Stockwerken; einige waren 21 Stockwerke hoch, andere nur ein Stockwerk. Ein schweres Spiel würde also eine mittlere Höhe von 18 Stockwerken haben. Denn dann würde das Flugzeug ziemlich schnell mit einem Gebäude zusammenstoßen und dann abstürzen. Sehr einfach wäre das Spiel z. B. bei einer Durchschnittshöhe von fünf Stockwerken. Das Flugzeug würde dann

ziemlich lange brauchen, bis es mit einem Wolkenkratzer zusammenstieße. Verständlicherweise wäre es nicht sinnvoll, bei einer Durchschnittshöhe von fünf plötzlich ein oder zwei Gebäude mit 20 Stockwerken dazwischen zu haben. Das Spiel wäre dann schon als schwierig einzustufen.

Zu Beginn wollen wir also den Spieler nach dem Schwierigkeitsgrad fragen (z. B. zwischen 1 und 9). Damit können wir Wolkenkratzer bauen, deren ungefähre Höhe von dem gewählten Schwierigkeitsgrad abhängt. Versuchen Sie einmal folgende Routine:

```
  10 INPUT "SCHWIERIGKEITSGRAD (
1-9)";d
  20 IF d<1 OR d>9 THEN GO TO 10
  30 LET d=12-d
  35 CLS
  40 FOR a=0 TO 31
  50 FOR b=d+INT (RND*d) TO 21
  60 PRINT AT b,a;"■"
  70 NEXT b
  80 NEXT a
  90 GO TO 10
```

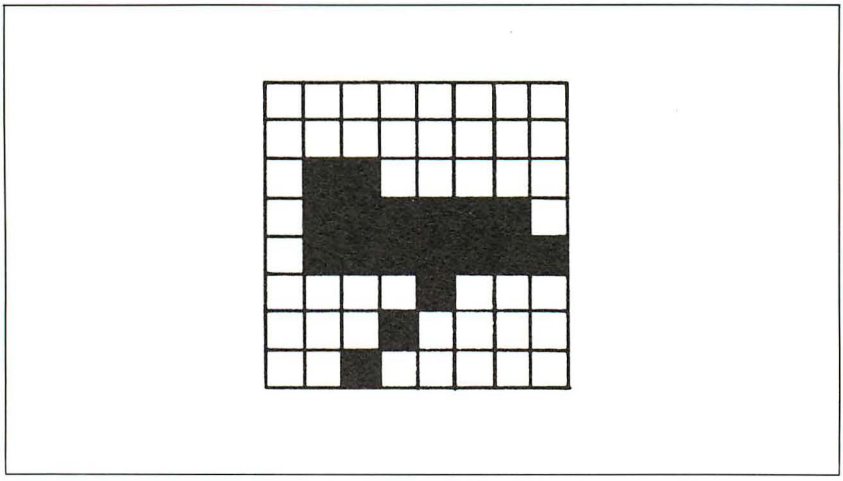
Geben Sie eine Zahl von 1 bis 9 ein. Der Spectrum zeichnet dann eine ziemlich realistische Silhouette. Probieren Sie eine kleine Zahl und Sie erhalten eine entsprechend flache Bebauung. Was passiert, wenn Sie eine Zahl außerhalb des zulässigen Bereichs eingeben? Warum? Jedes Programm, das Sie schreiben, sollte eine Routine haben, die eine ungültige Eingabe zurückweist.

Es kommt öfter vor, daß man sich bei der Eingabe vertippt. Wenn ein solcher Fehler nicht bemerkt wird, kann das zu einem Programmabbruch führen oder aber verwirrende Ergebnisse produzieren. Eine einzelne Zeile, welche den zulässigen Bereich einer Zahl prüft, schafft hier Abhilfe.

Ist Ihnen das Programm bis zu diesem Punkt klar? Dann gehen wir einen Schritt weiter: Wir stellen ein Flugzeug her, das über die Stadt hinwegfliegen kann. Zuerst überlegen wir uns, wie unser Flugzeug aussehen soll. Auf der Tastatur finden wir jedoch kein Symbol, das im entferntesten wie ein Flugzeug aussieht. Es sieht so aus, als ob wir uns mit Grafik befassen müssen. Hier ist ein Weg, wie wir Schritt für Schritt vorgehen können.

1. Zeichnen Sie ein Gitter mit einen 8x8-Raster.
2. Füllen Sie jetzt so lange Quadrate aus, bis das Ganze von der Seite her einem Flugzeug ähnelt. Keine Angst! Auch wenn es ein bißchen komisch aussieht, auf dem Bildschirm wird es wahrscheinlich prima aussehen.

Hier ist das Flugzeug, welches im Programm benutzt wird:



Jetzt gehen wir jede Reihe durch und schreiben eine Null für jedes nicht ausgefüllte und eine Eins für jedes ausgefüllte Quadrat. Wir erhalten dann folgende Tafel:

0 0 0 0 0 0 0 0	0
0 0 0 0 0 0 0 0	0
0 1 1 0 0 0 0 0	96
0 1 1 1 1 1 1 0	126
0 1 1 1 1 1 1 1	127
0 0 0 0 1 0 0 0	8
0 0 0 1 0 0 0 0	16
0 0 1 0 0 0 0 0	32

Jede Reihe ist eine binäre Zahl.

3. Mit Hilfe des Spectrum gehen wir jetzt die Reihe durch und tippen:

```
PRINT BIN (die Reihe der Nullen und Einsen)
```

Notieren Sie sich die Zahl, die der Computer angibt, direkt neben der binären Zahl. Wenn Sie z. B. die dritte Reihe tippen:

```
PRINT BIN 01100000
```

würde der Computer die Zahl 96 ausgeben. Schreiben Sie diese Zahl neben die entsprechende Reihe, also

```
0 1 1 0 0 0 0 0  96
```

Passen Sie auf, daß keine Null fehlt! Jede Null hat eine bestimmte Bedeutung. Wenn Sie z. B. rechts zwei Nullen vergessen und die binäre Zahl 011000 eingeben, erhalten Sie statt 96 nur die Zahl 24.

Sie müssen jetzt acht Zahlen haben. Im obigen Beispiel sind das der Reihenfolge nach: 0,0,9,126,127,8,16 und 32.

Löschen Sie jetzt den Spectrum (mittels NEW) und tippen Sie folgendes ein:

```
10 DATA ....ihre 8 zahlen ...
20 FOR a=0 TO 7
30 READ b: POKE USR "a"+a,b
40 NEXT a
```

Die DATA-Anweisung sollte die acht Zahlen enthalten, durch Kommata getrennt. Wenn Sie dies laufen lassen, rührt sich scheinbar nichts. Wenn Sie aber den Buchstaben a tippen, erscheint plötzlich ein winziges Flugzeug auf dem Bildschirm. Selbst wenn Sie jetzt das Programm löschen, bleibt doch das Flugzeug weiter einsatzbereit, es sei denn, Sie ziehen den Stecker heraus oder konstruieren sich ein anderes Flugzeug. Statt des Buchstabens a können Sie auch sagen: CHR$ 144. Nach dieser Methode können Sie sich beliebige andere Flugzeuge konstruieren.

Mit Hilfe des Flugzeugs können wir nun an die weiteren Aufgaben des Programms herangehen. Wir können das Flugzeug in Bewegung setzen und an Höhe verlieren lassen. Fügen Sie der letzten Häuserblockroutine noch die folgenden Zeilen hinzu (das Flugzeug steht in Zeile 140):

```
 90 LET u=0
100 LET p=0
140 PRINT AT u,p;CHR$ 144
160 LET p=p+1
170 PRINT AT u,p-1;" "
180 IF p=32 THEN LET p=0: LET u
=u+1: BEEP .1,u
190 GO TO 140
```

Sie werden nach dem Schwierigkeitsgrad gefragt. Die Stadt wird aufgebaut. Danach fliegt ein kleines Flugzeug quer über den Bildschirm und verliert von Mal zu Mal an Höhe. Dabei ertönt gleichzeitig ein BEEP. Die ansteigenden Töne sollen die Spannung erhöhen. Bald fliegt das Flugzeug durch das erste Gebäude hindurch. So senkt es sich langsam, bis es schließlich am unteren Rand des Bildschirms verschwindet. Danach stoppt das Programm mit einer Fehlermeldung.

Wir brauchen jetzt eine Zeile, die die jeweilige Position des Flugzeuges notiert und das Flugzeug explodieren läßt, wenn es auf einen Wolkenkratzer trifft. Dafür brauchen wir zwei Routinen: eine, um die jeweilige Position und den Zusammenprall zu ermitteln, und eine, die die Explosion darstellt.

Zuerst die Routine zur Überprüfung des Zusammenstoßes. Dafür gibt es zwei Möglichkeiten:

SCREEN$:

Ersetzen Sie Zeile 60 durch:

```
60  PRINT AT b,a; INVERSE 1; "X"
```

und Zeile 140 durch:

```
140  IF SCREEN$ (u,p) = "X" THEN STOP
```

SCREEN$ kann Buchstaben lesen. Daher stoppt das Programm, wenn nun das Flugzeug mit einem „X" zusammenstößt. In diesem Fall muß die Routine zum Zeichnen der Wolkenkratzer geändert werden. Statt Grafik-Blöcken müssen inverse „X" erzeugt werden, da SCREEN$ keine Grafik-Blöcke lesen kann.

ATTR:

Um die Attribut-Funktion zu verwenden, müssen wir Zeile 60 ersetzen durch:

```
60  PRINT AT, b,a; INK 5; PAPER 0; "Grafik-Block"
```

und Zeile 140 durch:

```
140  IF ATTR (u,p) = 5 THEN STOP
```

Mit diesem Befehl wird das Programm immer dann gestoppt, wenn das Flugzeug auf ein Bildschirmquadrat stößt, welches grün auf schwarz ist, nicht blinkt und extra hell erscheint. Der Nachteil dieser Methode liegt darin, daß sie ein wenig kompliziert ist. Wenn nämlich in einem Pro-

gramm eine Vielzahl verschiedener Farbflecke und Quadrate benutzt wird, wird es recht mühsam, für jede Position die Farbe und die Farbeigenschaften auszuarbeiten.

Soll das Flugzeug bei einem gelben, blinkenden Fleck abstoppen, müssen Sie Zeile 140 ändern:

```
140  IF ATTR (u,p) = 134 THEN STOP
```

Zur Erinnerung:

128,	falls das Quadrat blinkt (0, falls nicht)	1∗128 = 128
64,	falls das Quadrat besonders hell erscheint	0∗ 64 = 0
8∗	Papierfarbe (schwarz = 0)	8∗ 0 = 0
1∗	Farbe der Schrift (gelb = 6)	1∗ 6 = 6
	Summe:	134

Am verständlichsten ist sicherlich die Form mit dem Befehl SCREEN$:

```
PRINT AT b,a; INVERSE 1; "X"
IF SCREEN$ (u,p) = "X" THEN . . .
```

Da das Flugzeug nur zwei verschiedene Zeichen passiert, Himmel und Wolkenkratzer, genügt diese Form. Also:

```
140  IF SCREEN$ (u,p) = "X" THEN STOP
```

zusätzlich:

```
142  PRINT AT u,p; CHR$ 144
```

und vergessen Sie nicht:

```
60  PRINT AT b,a; INVERSE 1; "X"
```

Das Programm stoppt jetzt, wenn das Flugzeug auf ein Gebäude trifft.

Der bloße Befehl STOP ist etwas langweilig und phantasielos. Was wir brauchen, ist eine aufregende Explosion. Denn schließlich geht das Flugzeug zu Bruch. Versuchen Sie's mal damit:

```
200 LET x=p*8: LET y=(21-u)*8
205 FOR a=1 TO 40
207 PLOT x,y
210 DRAW INT (RND*256)-x,INT (R
ND*158)-y
220 BEEP .1,20: BEEP .01,10
230 NEXT a
```

Suchen Sie sich einen Punkt auf dem Bildschirm heraus und ordnen Sie ihm die Koordinaten u und p zu. Das sieht dann so aus: LET u=11: LET p=16. Geben Sie dies direkt ein und sagen Sie dann: GO TO 200.

Jetzt müßten Sie eigentlich sehen, wie der Punkt „explodiert"; natürlich mit entsprechendem Geräusch. Wir müssen nur noch Zeile 140 abändern:

```
140  IF SCREEN$ (u,p) = "X" THEN GO TO 200
```

Starten Sie das Programm mit RUN. Was wir jetzt sehen, ist:

a) eine auf dem Bildschirm gezeichnete Stadt
b) ein vorbeifliegendes Flugzeug, das tiefer und tiefer sinkt und schließlich
c) auf ein Gebäude trifft und explodiert.

Lesen Sie sich das Programm noch einmal durch. Haben Sie jeden Schritt verstanden? Dann gehen wir jetzt zum letzten Teil über. Es geht zunächst einmal um die Konstruktion einer Bombe. Eine solche Bombe können wir aus dem Flugzeug auf die Wolkenkratzer fallen lassen. Wenn die Wolkenkratzer dem Erdboden gleichgemacht werden, kann das Flugzeug bis unten hin weiterfliegen. Danach muß das Programm noch einen etwas professionelleren Anstrich bekommen.

Höchstwahrscheinlich werden Sie nie dazu kommen, diesem oder irgendeinem anderen Spiel den endgültig letzten Schliff zu geben. Denn kaum, daß Sie es mit SAVE gespeichert haben und stolz Ihr Werk bewundern, kommt schon jemand vorbei, spielt damit und gibt Ihnen dann die zündenden Ideen, wie Sie Ihr Spiel noch verbessern könnten.

Doch machen wir uns darüber keine Gedanken. Wir müssen jetzt den Bomben-Teil des Programms noch zusammenbasteln. Dafür benötigen wir einen Schalter. Üblicherweise haben Schalter nur die Werte 0 und 1, 0 für aus und 1 für an. In diesem Programm bezeichnen wir den Schalter mit s. Dieser Schalter soll den Wert 1 haben, wenn sich die Bombe in der Luft befindet, und sonst 0. Zu Beginn des Spiels hat der Schalter den Wert 0, da die Bombe nicht fällt. Also wird s anfangs auf 0 gesetzt, und das Programm sieht jetzt so aus:

```
  10 INPUT "SCHWIERIGKEITSGRAD (
1-9)",d
  20 IF d<1 OR d>9 THEN GO TO 10
  30 LET d=12-d
  35 CLS
  40 FOR a=0 TO 31
  50 FOR b=d+INT (RND*d) TO 21
  60 PRINT AT b,a; INVERSE 1;"X"
  70 NEXT b
  80 NEXT a
```

```
  90 LET u=0
 100 LET p=0
 110 LET s=0
 140 IF SCREEN$ (u,p)="X" THEN G
O TO 200
 142 PRINT AT u,p;CHR$ 144
 160 LET p=p+1
 170 PRINT AT u,p-1;" "
 180 IF p=32 THEN LET p=0: LET u
=u+1: BEEP .1,u
 190 GO TO 140
 200 LET x=p*8: LET y=(21-u)*8
 205 FOR a=1 TO 40
 207 PLOT x,y
 210 DRAW INT (RND*256)-x,INT (R
ND*158)-y
 220 BEEP .1,20: BEEP .01,10
 230 NEXT a
```

Die Bombe fällt nicht. Also ist der Schalter s aus (s=0). Wir wollen dem Spieler jetzt erlauben, die Bombe fallen zu lassen (s=1). Dazu brauchen wir nur eine zusätzliche Zeile:

```
148 IF INKEY$ <> " " THEN LET s=1
```

INKEY$ enthält das aktuell auf der Tastatur eingetippte Zeichen (wenn nichts getippt wird, ist INKEY$ leer, d. h. = " "). Der Schalter s wird also auf 1 gesetzt, sobald irgendeine Taste getippt wird. Durch eine geringfügige Änderung fällt die Bombe nur, wenn die 0-Taste gedrückt wird:

```
148 IF INKEY$ = "0" THEN LET s=1
```

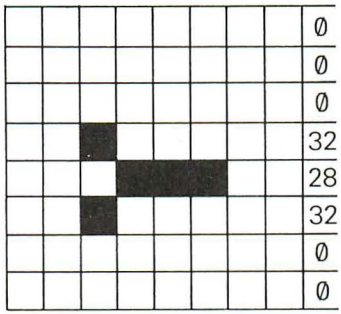
Es ist aber in jedem Fall einfacher, wenn Sie mit jeder beliebigen Taste den Bombenwurf auslösen können. Lassen wir also Zeile 148 in der ersten Form.

Als nächstes benötigen wir zwei Variablen, die die Position der Bombe auf dem Bildschirm notieren. Solange die Bombe nicht ausgeklinkt ist, wartet sie im Inneren des Flugzeuges (Schalter s=0):

```
144 IF s = 0 THEN LET a=u: LET t=p (a und t sind die Koordinaten der
Bombe)
```

Wir benötigen eine Figur, die wie eine Bombe aussieht. Ein einfacher Punkt würde genügen, aber vielleicht wollen Sie Ihre eigene Bombe entwerfen.

Versuchen Sie das doch mit der oben angegebenen Methode. Eine mögliche Figur ist die folgende:



Und so sieht die Zeile aus, mit der das Zeichen festgelegt wird:

```
POKE USR "b" + a,b
```

und nicht:

```
POKE USR "a" + a,b
```

denn sonst würde das Flugzeug verloren gehen.

Um die Figur der Bombe einzusetzen, können Sie wahlweise das Grafik-b oder CHR$ 145 verwenden. Wenn Sie den Bombenabwurf mit einem Punkt durchführen wollen, setzen Sie überall, wo CHR$ 145 steht, einen Punkt ein.

Wenn wir die Figur für die Bombe fertig haben, können wir darangehen, die Bombe auf dem Bildschirm darzustellen. Da die Position der Bombe anfangs mit der Position des Flugzeuges identisch ist (s=0), zeichnen wir sie erst, wenn sich die Bombe vom Flugzeug löst:

```
158  IF s=1 THEN PRINT AT a,t; CHR$ 145: BEEP .01,60 - a
```

Die Bombe erscheint jetzt mit jedem x-beliebigen Tastendruck auf dem Bildschirm. Sie fällt aber erst dann herunter, wenn wir die folgende Zeile einfügen:

```
155  IF s=1 THEN PRINT AT a,t;" ": LET a=a+1
```

Die Bombe bewegt sich jetzt abwärts, weil mit der Variablen a auch die Koordinaten für die Position der Bombe verändert werden.

Wir haben zwar jetzt eine fallende Bombe; das ist aber noch nicht genug, denn das Programm bricht mit einer Fehlermeldung ab, sobald die Bombe unten angekommen ist. Wir brauchen eine Programmzeile, in der geprüft wird, ob die Bombe ein Gebäude getroffen hat (ähnlich Zeile 140) oder am Boden angekommen ist.

```
157  IF SCREEN$ (a,t) = "X" OR a=21 THEN GO TO 300
```

Nun können wir noch eine Routine formulieren, mit der ein Haus in die Luft gesprengt wird:

```
300  FOR a=a TO 21
310  IF RND > .99 THEN GO TO 340
315  BEEP .005,a-20
320  PRINT AT a,t; " "
330  NEXT a
340  LET s=0: GO TO 140
```

Was wir jetzt noch benötigen, ist:

```
235  GO TO 10
```

Jetzt haben wir ein Spiel, was ohne Unterbrechung durchläuft. Es ist ziemlich schwierig, die Bombe gezielt auf ein Gebäude fallen zu lassen, das zerstört werden soll.

Und so sieht das vollständige Programm jetzt aus:

```
  10 INPUT "SCHWIERIGKEITSGRAD (
1-9)";d
  20 IF d<1 OR d>9 THEN GO TO 10
  30 LET d=12-d
  35 CLS
  40 FOR a=0 TO 31
  50 FOR b=d+INT (RND*d) TO 21
  60 PRINT AT b,a; INVERSE 1;"X"
  70 NEXT b
  80 NEXT a
  90 LET u=0
 100 LET p=0
 110 LET s=0
 140 IF SCREEN$ (u,p)="X" THEN G
O TO 200
 142 PRINT AT u,p;CHR$ 144
 144 IF s=0 THEN LET a=u: LET t=
p
```

```
 148 IF INKEY$<>"" THEN LET s=1
 155 IF s=1 THEN PRINT AT a,t;"
   LET a=a+1
 157 IF SCREEN$ (a,t)="X" OR a=2
1 THEN GO TO 300
 158 IF s=1 THEN PRINT AT a,t;CH
R$ 145; BEEP .01,60-a
 160 LET p=p+1
 170 PRINT AT u,p-1;" "
 180 IF p=32 THEN LET p=0: LET u
=u+1: BEEP .1,u
 190 GO TO 140
 200 LET x=p*8: LET y=(21-u)*8
 205 FOR a=1 TO 40
 207 PLOT x,y
 210 DRAW INT (RND*256)-x,INT (R
ND*158)-y
 220 BEEP .1,20: BEEP .01,10
 230 NEXT a
 235 GO TO 10
 300 FOR a=a TO 21
 310 IF RND>.99 THEN GO TO 340
 315 BEEP .005,a-20
 320 PRINT AT a,t;" "
 330 NEXT a
 340 LET s=0: GO TO 140
```

Das Spiel funktioniert zwar, aber einiges fehlt noch: Es hat weder Farbe noch einen Punktezähler. Versuchen Sie bitte selbst, diese Dinge einzubauen. Nehmen Sie eine Variable für den Punktezähler, der die Zahl der zerstörten Häuser festhält. Am Spielende muß dann der Punktestand angegeben werden. In der augenblicklichen Fassung am Ende dieses Abschnitts gibt es zusätzlich einen Zähler, der den höchsten Punktstand über mehrere Spiele notiert. Solche geringfügigen Änderungen lassen sich leicht einbauen und bringen gleich eine wesentliche Verbesserung des Programms. Wenn Sie jetzt noch Wolkenkratzer mit Fenstern statt mit X bauen, werden Sie sicher überrascht sein.

In der endgültigen Fassung sind auch die DATA-Anweisungen für die verschiedenen Grafik-Zeichen enthalten. Jetzt kann das Programm komplett eingegeben und gestartet werden.

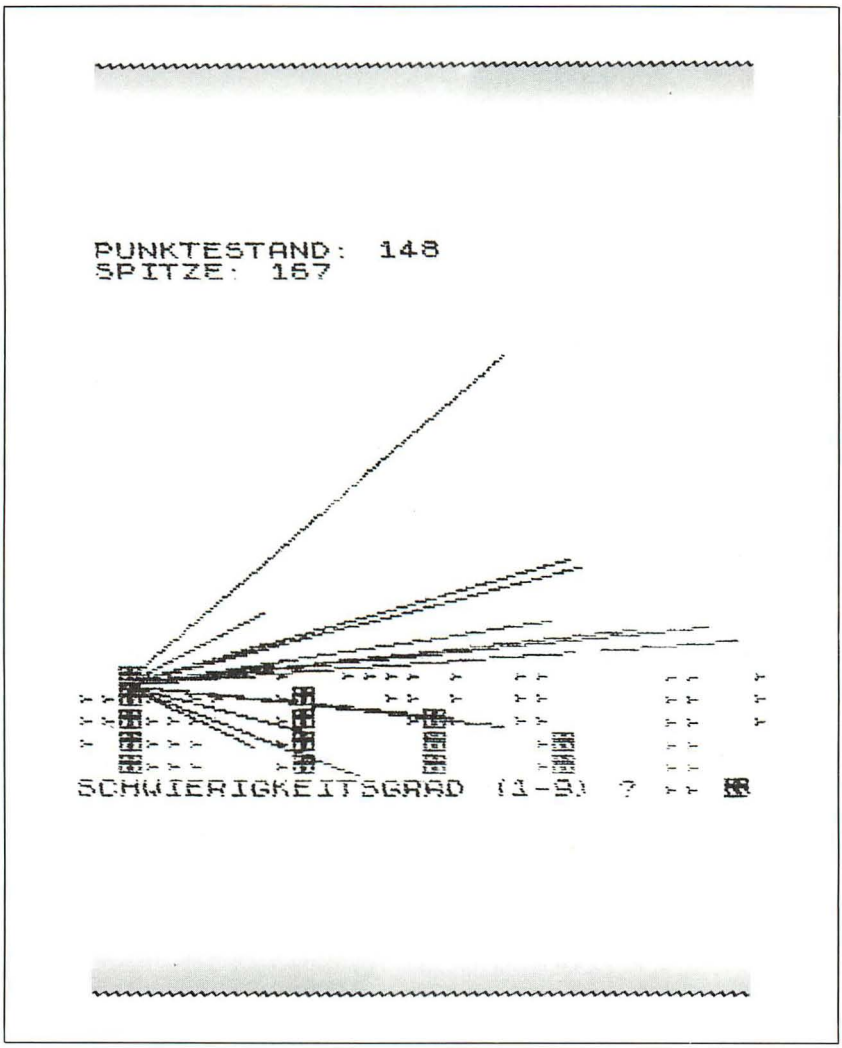
Farbversion von NÄCHTLICHER ÜBERFALL:

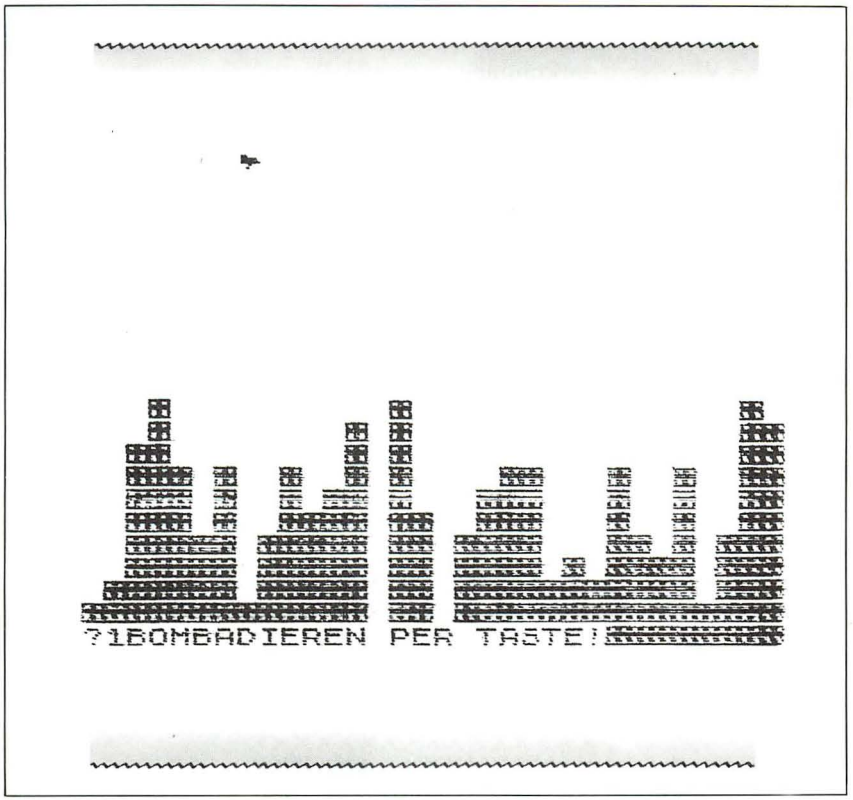
```
   2 LET h=0
   4 IF h=0 THEN GO SUB 1000
   5 LET punkte=0
  10 PAPER 0: INK 6: BORDER 1
  30 PRINT AT 21,0,"SCHWIERIGKEI
TSGRAD (1-9) ?": LET d$=INKEY$:
IF LEN d$<>1 OR CODE d$<49 OR CO
DE d$>57 THEN GO TO 30
```

```
   32 CLS
   35 LET d=12-VAL d$
   40 FOR a=0 TO 31
   50 FOR b=d+INT (RND*d) TO 21
   60 PRINT PAPER 0;AT b,a;CHR$ 1
46
   70 NEXT b
   80 NEXT a
   90 LET u=0
  100 LET p=0
  110 LET s=0
  130 PRINT AT 21,0;d$+" BOMBADIE
REN PER TASTENDRUCK!"
  140 PRINT AT u,p;: IF PEEK (PEE
K 23684+256*peek 23685)=255 THEN
 GO TO 200
  142 PRINT INK 4;CHR$ 144
  144 IF s=0 THEN LET a=u: LET t=
p
  148 IF INKEY$<>"" THEN LET s=1
  155 IF s=1 THEN PRINT AT a,t;"
": LET a=a+1: IF a=22 THEN GO TO
 340
  157 PRINT AT a,t;: IF PEEK (PEE
K 23684+256*PEEK 23685)=255 THEN
 GO TO 300
  158 IF s=1 THEN PRINT INK 2;CHR
$ 145: BEEP .01,60-a
  160 LET p=p+1
  170 PRINT AT u,p-1;" "
  180 IF p=32 THEN LET p=0: LET u
=u+1: BEEP .1,u
  185 IF u=22 THEN GO TO 1200
  190 GO TO 140
  200 LET x=p*8: LET y=(21-u)*8
  205 FOR a=1 TO punkte STEP 10
  207 PLOT x,y
  210 DRAW INK 6;INT (RND*256)-x,
INT (RND*158)-y
  220 BEEP .1,20: BEEP .01,10
  230 NEXT a
  235 GO TO 400
  300 FOR a=a TO 21
  310 IF RND>.99 THEN GO TO 340
  315 BEEP .005,a-20
  320 PRINT AT a,t;" "
  325 LET punkte=punkte+1: IF pun
kte/250=INT (punkte/250) THEN PO
KE 23624,PEEK 23624+8
  330 NEXT a
  340 LET s=0: GO TO 140
  400 IF h<punkte THEN LET h=punk
te
  410 PRINT AT 0,0;"PUNKTESTAND:
";punkte,"SPITZE: ";h
  420 GO TO 5
1000 DATA "a",0,0,96,126,127,8,1
6,0,"b",0,0,0,32,28,32,0,0,"c",2
55,153,153,255,255,153,153,255
1005 FOR s=1 to 3
1007 READ a$
1010 FOR a=0 TO 7
1020 READ b: POKE USR a$+a,b
1030 NEXT a
1045 NEXT s
1050 RETURN
1200 DATA 0,4,7
1210 FOR a=1 TO 3
1215 READ c
```

```
1220 FOR b=1 TO 3
1230 BEEP 1/3,c
1240 NEXT b
1250 NEXT a
1260 BEEP 1,12
1270 GO TO 32
```

Ausschnitte aus dem Spielverlauf:





Wie wir anhand der ausführlichen Erläuterungen zu diesem Spiel gesehen haben, ist es gar nicht so schwer, solche Spiele zu programmieren. Wir müssen nur ähnlich systematisch vorgehen und die Programme Schritt für Schritt zusammenbauen. Die nun folgenden Programme sind nach einem ähnlichen Schema aufgebaut. Die Erläuterungen wurden daher etwas kürzer gefaßt.

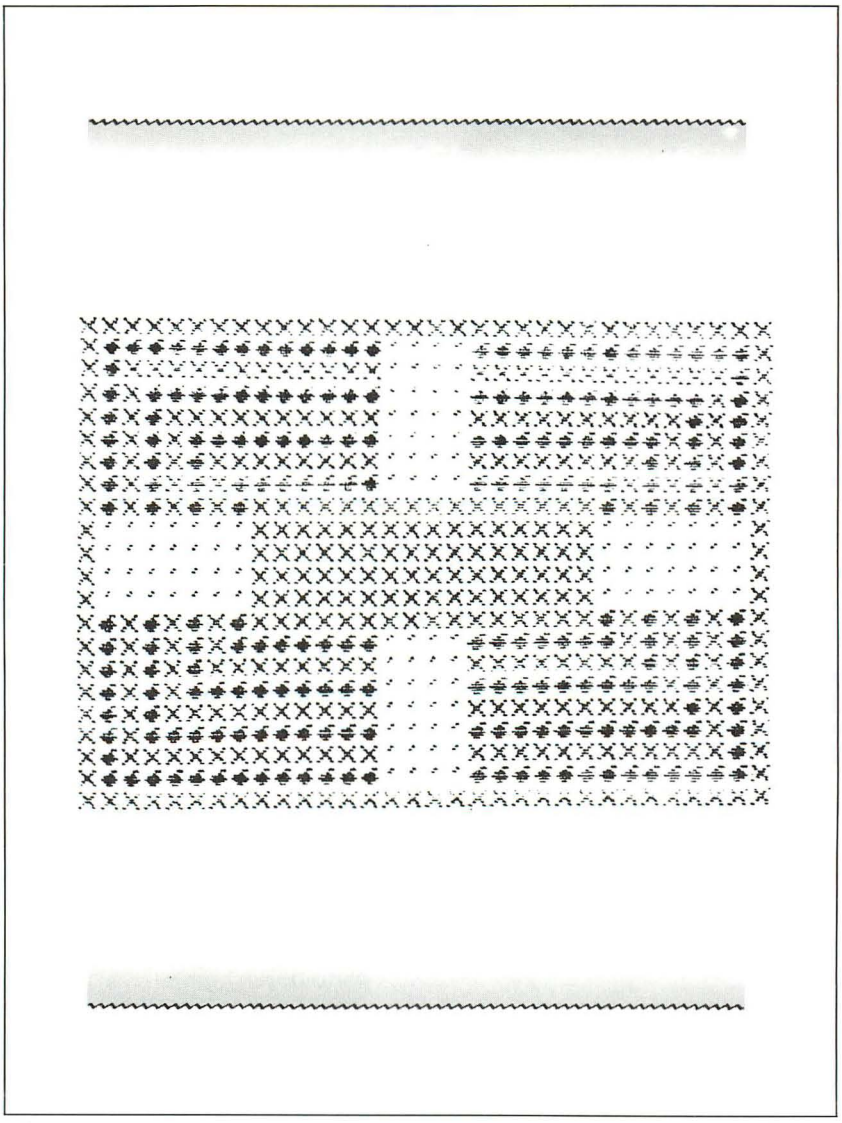
Rebenklau

Dieses Programm lehnt sich an die Vorbilder DODGEM und PACMAN an. Darin steuert der Spieler ein kleines Auto durch einen Irrgarten und versucht dabei, Zusammenstöße mit einem anderen Auto zu vermeiden, das vom Computer gelenkt wird.

Im REBENKLAU ist die Szenerie weniger grausig. Jacques ist ein Weintraubendieb in einem gepflegten französischen Weingarten. Ziel des

Spiels ist es, Jacques im Weingarten herumzuführen und ihn möglichst viele Trauben verdrücken zu lassen. Gleichzeitig muß er versuchen, dem wütenden Winzer zu entkommen.

So sieht der Weingarten aus, wenn die Trauben noch alle da sind:



Die Routine, mit der der Garten mit seinen farbigen Mauern und den Trauben gezeichnet wird, macht fast ein Drittel des ganzen Programms aus (alle Trauben haben die gleiche Farbe, aber eine andere als die Mauern). Es empfiehlt sich, die Routine unten mit SAVE zu speichern, damit Sie später noch etwas ergänzen können. Dann müssen Sie später nicht mehr alles von vorn eintippen.

Achten Sie bitte beim Tippen darauf, daß die G's im Grafik-Modus eingegeben werden.

```
    2 LET h=0
    5 LET e$="Jacques"
   10 DATA 60,126,240,224,224,240
,126,60
   12 DATA 0,66,195,195,231,255,1
26,60,60,126,15,7,7,15,126,60
   14 DATA 60,126,255,231,195,195
,66,0,0,0,0,0,60,60,60,60
   18 DATA 60,126,255,255,255,255
,126,60,0,6,8,60,126,60,24,0
   20 DATA 24,60,24,255,24,24,36,
102
   25 FOR a=0 TO 7
   30 FOR b=0 TO 7
   40 READ c
   50 POKE USR CHR$ (144+a)+b,c
   60 NEXT b
   70 NEXT a
   75 LET s=0: LET w=1
   77 LET z=4: LET y=0: PAPER 3:
INVERSE 1
   78 LET f=0
   79 CLS
   80 PRINT INK z;"XXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXX"
   90 IF f=1 THEN RETURN
  100 PRINT INK z;"X"; INK y;"GGG
GGGGGGGGGG''''GGGGGGGGGGGGG"; IN
K z;"X"
  110 IF f=1 THEN GO TO 80
  120 PRINT INK z;"X"; INK y;"G";
 INK z;"XXXXXXXXXXX"; INK y;"''
''"; INK z;"XXXXXXXXXXX"; INK y
;"G"; INK z;"X"
  130 IF f=1 THEN GO TO 100
  140 PRINT INK z;"X"; INK y;"G";
 INK z;"X"; INK y;"GGGGGGGGGG''
''GGGGGGGGGG"; INK z;"X"; INK y
;"G"; INK z;"X"
  150 IF f=1 THEN GO TO 120
  160 PRINT INK z;"X"; INK y;"G";
 INK z;"X"; INK y;"G"; INK z;"XX
XXXXXXX"; INK y;"''''"; INK z;"
XXXXXXXXX"; INK y;"G"; INK z;"X
"; INK y;"G"; INK z;"X"
  170 IF f=1 THEN GO TO 140
  180 PRINT INK z;"X"; INK y;"G";
 INK z;"X"; INK y;"G"; INK z;"X"
; INK y;"GGGGGGGGG''''GGGGGGGGG"
; INK z;"X"; INK y;"G"; INK z;"X
"; INK y;"G"; INK z;"X"
  190 IF f=1 THEN GO TO 160
  200 PRINT INK z;"X"; INK y;"G";
```

```
  INK z;"X"; INK y;"G", INK z;"X"
, INK y;"G"; INK z;"XXXXXXXX", I
NK y;"´´´´"; INK z;"XXXXXXXX", I
NK y,"G", INK z,"X", INK y,"G",
INK z,"X"; INK y,"G", INK z,"X"
  210 IF f=1 THEN GO TO 180
  220 PRINT INK z,"X", INK y,"G";
  INK z,"X", INK y;"G"; INK z,"X"
, INK y;"G"; INK z;"X"; INK y,"G
GGGGGG´´´´GGGGGGG", INK z,"X", I
NK y;"G"; INK z,"X"; INK y,"G",
INK z;"X"; INK y;"G", INK z,"X"
  230 IF f=1 THEN GO TO 200
  240 PRINT INK z;"X"; INK y;"G";
  INK z;"X"; INK y;"G"; INK z;"X"
; INK y;"G"; INK z;"X"; INK y;"G
", INK z;"XXXXXXXXXXXXXXXX"; INK
 y;"G"; INK z;"X"; INK y;"G"; IN
K z;"X"; INK y;"G"; INK z;"X"; I
NK y;"G"; INK z;"X"
  250 IF f=1 THEN GO TO 220
  260 FOR a=1 TO 4
  270 PRINT INK z;"X"; INK y;"´´´
´´´´"; INK z;"XXXXXXXXXXXXXXXX";
  INK y;"´´´´´´´´"; INK z;"X"
  280 NEXT a
  290 LET f=1
  300 GO SUB 240
  305 INVERSE 0
  307 GO SUB 1040
  310 LET u=20
  315 PAPER 0: INK 6
  320 LET p=17
  330 LET du=0
  335 LET lj=4
```

Nachdem Sie den Punktehöchststand sowie die übrigen Variablen auf ihren Startwert gesetzt (siehe die Variablenliste am Ende) und den Weingarten gezeichnet haben, springt das Programm zu einer Routine, die

1. die „Supertraube" zeichnet, eine blinkende Frucht. Wer sie ißt, erhält fünf Punkte statt wie üblich einen Punkt pro Traube. Sobald Jacques oder der Winzer die Traube ißt, springt das Programm zu einem anderen Teil des Weingartens;

2. eine kleine Melodie spielt. Viele Automatenspiele starten mit einer Erkennungsmelodie. Nach mehreren Spielen geht einem diese Melodie allerdings auf die Nerven. Wenn Sie also die Musik stört, lassen Sie die Melodie ruhig weg. Die Noten für die Melodie befinden sich in der DATA-Anweisung in Zeile 1050.

Da das Programm generell zu dieser Routine springt, sobald der Weingarten leergefegt ist, wird der Punktehöchststand geändert, wenn der aktuelle Punktestand darüber liegt.

Bei der Rückkehr aus diesem Unterprogramm werden die meisten Variablen wieder auf ihren Anfangswert gesetzt. Das Programm setzt dann fort

mit dem Hauptteil des Programms, welcher die einzelnen Aktionen steuert. Von hier aus geht das Programm zu einer Routine, die Jacques durch den Weingarten bewegt; von hier aus wird auch die entsprechende Routine aufgerufen, wenn Jacques vor eine Mauer läuft.

Es gibt folgende Unterprogramme:

a) *Automatische Bewegungssteuerung für Jacques*

Sie ändert die Richtung und die Figur, wenn Jacques an eine Mauer stößt. Wenn er beispielsweise in östlicher Richtung geht und auf eine Mauer trifft, ändert das Programm automatisch seine Richtung nach Norden, und auch die Figur paßt sich dieser Richtung an. Der Spieler hat keinen Einfluß darauf. Jacques bewegt sich ständig gegen den Uhrzeigersinn.

b) *Routine, mit der Jacques' Bewegungen von Hand gesteuert werden*

Jede Traubenreihe ist durch eine Mauer abgetrennt, so daß Jacques nicht quer zu den Reihen laufen kann. Es gibt jedoch vier freie Felder, wo Jacques die Reihen wechseln und dem heranstürmenden Winzer entkommen kann. Diese Routine wird benutzt, wenn

1. Jacques sich auf einem freien Feld befindet und
2. der Spieler eine Taste drückt.

Es wird geprüft, ob der Spieler Jacques in die richtige Richtung lenken möchte. Wenn ja, bewegt sich Jacques in diese Richtung. Die entsprechenden Variablen werden angepaßt und ein Ton erklingt, um dem Spieler akustisch zu helfen. Wird nämlich irgend etwas falsch eingegeben, erklingt ein anderer Ton.

c) *Routine zur Bewegung des Winzers*

Mit dieser Routine wird die Richtung des Winzers korrigiert, sobald dieser auf eine Mauer trifft (siehe Routine a).

d) *Routine, mit der der Winzer von Hand gesteuert wird*

Diese Routine wird immer dann aufgerufen, wenn

1. der Winzer auf einem der vier freien Felder steht und
2. der Winzer sich nicht in der gleichen Reihe wie Jacques befindet.

Der Winzer ist also ständig bestrebt, einen Zusammenprall mit Jacques herbeizuführen. Am Anfang kann er jedoch beim Wechsel immer nur eine Reihe weiter springen. Wenn Jacques aber mehr und mehr Trauben beseitigt, nimmt die Beweglichkeit des Winzers zu und das Spiel wird immer schwieriger.

e) *Routine zum Löschen des Spielfeldes*

Diese Routine wird gebraucht, wenn Jacques alle Trauben aufgegessen hat. Dann wird ein neuer Weingarten angelegt. Diese Routine besteht im Prinzip aus einer Reihe von Befehlen, die selbst wiederum weitere Unterroutinen aufrufen.

f) *Routine zum Spielende*

Diese Routine wird aufgerufen, wenn der Winzer Jacques schnappt. Jacques wird, mit entsprechender musikalischer Untermalung, „zur Schnecke gemacht". Falls nötig, wird der Punktehöchststand verändert. Hierzu wird der Spieler nach seinem Namen gefragt. Danach springt das Programm zurück an den Anfang, ohne jedoch dabei den Punktehöchststand zu löschen.

g) *Routine zum Zeichnen der Supertraube*

Diese Routine wird zu Programmbeginn aufgerufen bzw. dann, wenn Jacques diese Traube aufgegessen hat. Die alte Supertraube wird gelöscht (durch Zurücksetzen von I$). Danach wird ein geeigneter Platz für eine neue Supertraube auf dem Bildschirm gesucht. Falls dieser zufällig gewählte Platz ungeeignet ist, sucht das Programm einen anderen Platz. Die Auswahl des Platzes unterliegt dabei dem Zufall.

Was ist nun ein geeigneter Platz?

1. ein Punkt, der nicht zur Mauer gehört
2. ein Punkt, der nicht im freien Feld liegt
3. ein Punkt, wo sich bereits eine normale Traube befindet
4. ein Punkt, an dem Jacques bereits eine Traube gegessen hat, falls mehr als die Hälfte gegessen sind.

Das Hauptprogramm setzt Jacques und den Winzer in ihrer jeweiligen Richtung in Bewegung und ruft bei Bedarf die oben vorgestellten Routinen auf. Diese Routinen sind folgendermaßen angeordnet:

	Zeilen
a) Jacques automatisch bewegen	1000–1015
b) Jacques von Hand bewegen	565– 700
c) den Winzer automatisch bewegen	1020–1030
d) den Winzer von Hand bewegen	710– 790
e) Ende des Spiels	800– 900
f) Supertraube zeichnen	1150–1250

Zunächst sind diese Routinen erstellt worden, und danach erst wurde das Spiel als Ganzes vervollständigt.

Vielleicht ist es für Sie einfacher, das Programm einzugeben, wenn Ihnen ein Freund die Programmzeilen diktiert (vor allem die Routinen für den Weingarten mit den vielen INK- und PAPER-Befehlen). Beim Tippen ist es sehr wichtig, das Grafik-G zu verwenden und nicht das normale G. Sonst werden die Trauben nicht ausgedruckt. Vielleicht möchten Sie erst den Teil des Programms eintippen, der die frei definierbaren grafischen Zeichen festlegt. Dann erscheint bei jeder Eingabe des grafischen G eine Weintraube auf dem Bildschirm.

Endgültige Fassung des Programms REBENKLAU:

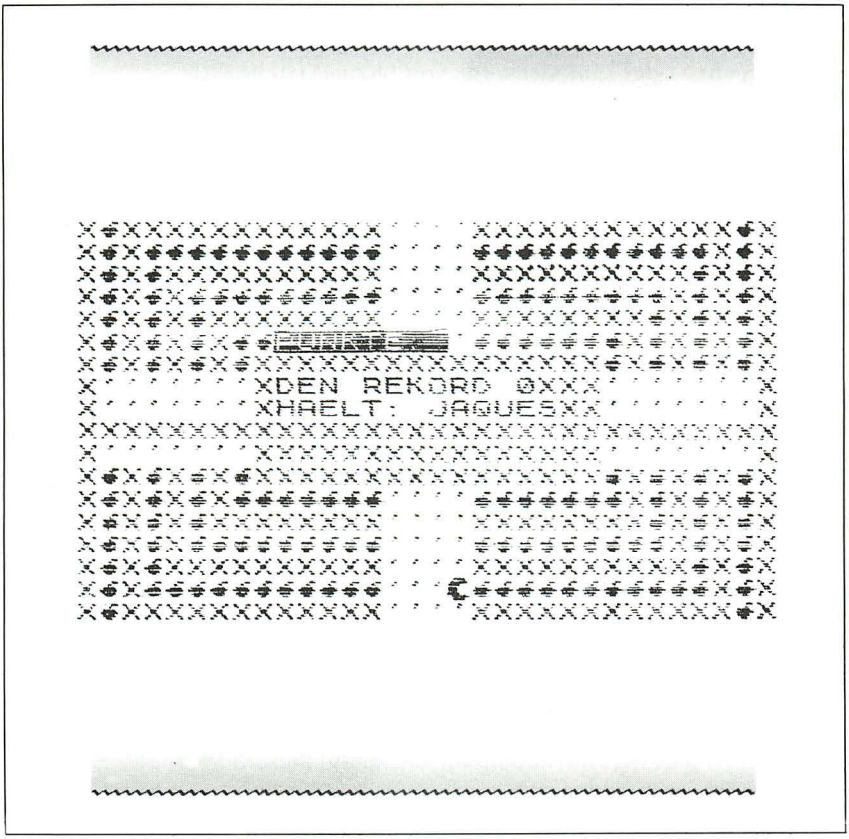
```
    2 LET h=0
    5 LET e$="JACQUES"
   10 DATA 60,126,240,224,224,240
,126,60
   12 DATA 0,66,195,195,231,255,1
26,60,60,126,15,7,7,15,126,60
   14 DATA 60,126,255,231,195,195
,66,0,0,0,0,0,0,60,60,60,60
   16 DATA 60,126,255,255,255,255
,126,60,0,6,8,60,126,60,24,0
   20 DATA 24,60,24,255,24,24,36,
102
   25 FOR a=0 TO 7
   30 FOR b=0 TO 7
   40 READ c
   50 POKE USR CHR$ (144+a)+b,c
   60 NEXT b
   70 NEXT a
   75 LET s=0: LET v=1: LET w=1
   77 LET z=4: LET y=0: PAPER 3:
INVERSE 1
   78 LET f=0
   79 CLS
   80 PRINT INK z;"XXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXX"
   90 IF f=1 THEN RETURN
  100 PRINT INK z;"X"; INK y;"GGG
GGGGGGGGGG''''GGGGGGGGGGGGG"; IN
K z;"X"
  110 IF f=1 THEN GO TO 80
  120 PRINT INK z;"X"; INK y;"G";
 INK z;"XXXXXXXXXXXX"; INK y;"'
'"; INK z;"XXXXXXXXXXXX"; INK y
;"G"; INK z;"X"
  130 IF f=1 THEN GO TO 100
  140 PRINT INK z;"X"; INK y;"G";
 INK z;"X"; INK y;"GGGGGGGGGG''
''GGGGGGGGGG"; INK z;"X"; INK y
;"G"; INK z;"X"
  150 IF f=1 THEN GO TO 120
  160 PRINT INK z;"X"; INK y;"G";
 INK z;"X"; INK y;"G"; INK z;"XX
XXXXXXX"; INK y;"''''"; INK z;"
XXXXXXXXX"; INK y;"G"; INK z;"X
"; INK y;"G"; INK z;"X"
  170 IF f=1 THEN GO TO 140
  180 PRINT INK z;"X"; INK y;"G";
 INK z;"X"; INK y;"G"; INK z;"X"
; INK y;"GGGGGGGGG''''GGGGGGGGG"
```

```
, INK z;"X"; INK y;"G"; INK z;"x
", INK y;"G"; INK z "X"
 190 IF f=1 THEN GO TO 160
 200 PRINT INK z;"X"; INK y;"G";
 INK z;"x"; INK y;"G"; INK z;"x"
, INK y;"G"; INK z;"XXXXXXXX"; I
NK y;"''''"; INK z;"XXXXXXXX"; I
NK y;"G"; INK z;"X"; INK y,"G",
INK z;"X"; INK y;"G", INK z,"x"
 210 IF f=1 THEN GO TO 180
 220 PRINT INK z;"X"; INK y,"G",
 INK z;"X"; INK y;"G"; INK z,"x"
, INK y;"G"; INK z;"X"; INK y,"G
GGGGGG''''GGGGGGG", INK z,"X", I
NK y,"G"; INK z;"X"; INK y,"G",
INK Z,"X"; INK y;"G"; INK z,"x"
 230 IF f=1 THEN GO TO 200
 240 PRINT INK z;"X"; INK y;"G",
 INK z;"X"; INK y;"G", INK z,"x"
, INK y;"G"; INK z;"X", INK y."G
"; INK z;"XXXXXXXXXXXXXXXX"; INK
 y;"G"; INK z;"X"; INK y;"G"; IN
K z;"X"; INK y;"G"; INK z;"X"; I
NK y;"G"; INK z;"X"
 250 IF f=1 THEN GO TO 220
 260 FOR a=1 TO 4
 270 PRINT INK z;"X"; INK y;"''''
''''"; INK z;"XXXXXXXXXXXXXXXX";
 INK y;"''''''''"; INK z;"X"
 280 NEXT a
 290 LET f=1
 300 GO SUB 240
 305 INVERSE 0
 307 GO SUB 1040
 310 LET u=20
 315 PAPER 0: INK 6
 320 LET p=17
 330 LET du=0
 335 LET lj=4
 340 LET dp=1
 345 LET lf=4
 350 LET a=10
 355 LET g$=" "
 360 LET t=1
 365 LET l$=" "
 370 LET da=-1
 375 PRINT AT 9,9; PAPER 0; INK
4; INVERSE 1;"PUNKTE: "
 380 LET dt=0
 385 LET q=1
 390 LET c=144
 400 IF INKEY$<>"" THEN BEEP .01
,0
 410 PRINT AT u,p; INK 6; CHR$ c
 440 IF SCREEN$ (u+du,p+dp)="X"
THEN GO TO 1000
 460 PRINT AT a,t; INK 0;g$
 470 IF SCREEN$ (a+da,t+dt)="X"
THEN GOT TO 1020
 490 LET g$=SCREEN$ (a+da,t+dt)
 495 IF ATTR (a+da,t+dt)>128 THE
N LET g$=","
 500 IF g$<>" " THEN LET q=v
 510 IF g$<>" " AND g$<>"," THEN
LET g$="G"
 515 LET a=a+da: LET t=t+dt
 517 IF a=ba AND t=rt THEN GO SU
B 1200
```

```
 520 IF SCREEN$ (a,t)="'" AND lj
<>lf THEN GO SUB 710
 525 PRINT AT a,t;"H"
 530 PRINT AT u,p; INK 3;l$
 540 LET u=u+du: LET p=p+dp
 542 IF ATTR (u,p)=6 THEN GO TO
800
 545 LET m=0: LET l$=SCREEN$ (u,
p): IF l$="" THEN LET l$=",": LE
T m=1
 547 IF u=ba AND p=rl THEN GO SU
B 1150
 550 PRINT AT u,p;"F"
 555 IF m=1 THEN LET s=s+1: LET
s2=s2+1: BEEP .005,-10: BEEP .00
5,-5: PRINT AT 9,17;s
 557 IF s2>=224 THEN GO TO 1110
 560 IF INKEY$="" OR i$="," THEN
 GOTO 400
 565 LET i$=INKEY$
 570 IF CODE INKEY$<53 OR CODE I
NKEY$>56 THEN GO TO 400
 580 RESTORE 585
 585 DATA du,"6","7",dp,"5","8"
 590 FOR i=1 TO 2
 600 READ j: READ j$: READ k$
 610 IF j=0 AND INKEY$<>j$ AND I
NKEY$<>k$ THEN GO TO 400
 620 NEXT i
 630 LET u1=u: LET p1=p
 640 LET u1=u+((i$="6")-(i$="7")
)*(du=0)*2
 650 LET p1=p+((i$="8")-(i$="5")
)*(dp=0)*2
 655 LET n=(i$="6")*(dp=1)+(i$="
7")*(dp=-1)+(i$="5")*(du=1)+(i$=
"8")*(du=-1)
 660 IF n=0 THEN LET n=-1
 665 IF lj+n=0 OR lj+n=5 THEN GO
TO 400
 670 LET lj=lj+n
 675 PRINT AT u,p; INK 3;l$
 680 LET u=u1: LET p=p1
 690 PRINT AT u,p;"F"
 695 BEEP .01,10
 700 GO TO 400
 710 IF INT q=0 THEN RETURN
 715 LET tf=lf+(lf<lj)-(lf>lj)
 720 LET q=q-1
 730 LET o=(lf>lj)*(da=-1)+(lf<l
j)*(da=1)
 740 IF o=0 THEN LET o=-1*(dt=0)
 745 LET o=o*2
 750 LET t=t+o
 760 LET o=(lf>lj)*(dt=1)+(lf<lj
)*(dt=-1)
 765 LET lf=tf
 770 IF o=0 THEN LET o=-1*(da=0)
 775 LET o=o*2
 780 LET a=a+o
 790 RETURN
 800 RESTORE 900
 805 FOR a=1 TO 4
 810 READ b$: READ r
 820 PRINT AT u,p;b$
 830 BEEP 1,r
```

```
 840 NEXT a
 850 FOR a=1 TO 64
 860 BEEP .01,25
 870 NEXT a
 872 PRINT AT 17,11;"SPIELENDE"
 875 IF h<s THEN INPUT "SIE HABE
N DIE HOECHSTPUNKTZAHL  ERREICHT
:BITTE GEBEN SIE IHREN  NAMEN UN
D ENTER EIN.";e$: IF LEN e$>11 T
HEN PRINT AT U,p-5;"ZU LANG.....
.": GOTO 800
 880 IF h<s THEN LET h=s
 885 IF INKEY$="" THEN GO TO 885
 890 GO TO 75
 900 DATA "F",30,"E",20,".",10,"
 ",0
1000 IF dp=0 THEN LET dU=-dp: LE
T dp=0: LET c=c+1: GO TO 420
1010 IF dp=0 THEN LET dp=dU: LET
 dU=0: LET c=c+1: IF c=148 THEN
LET c=144
1015 GO TO 420
1020 IF da=0 THEN LET da=dt: LET
 dt=0: GO TO 515
1030 IF dt=0 THEN LET dt=-da: LE
T da=0: GO TO 515
1040 RESTORE 1050
1042 LET s2=0
1045 GO SUB 1200
1050 DATA 7,1,7,.5,7,.5,10,1,12,
1,14,.5,12,1.5,10,1.5,12,.5,7,1,
7,.5,7,.5,10,1,12,1,7,2
1055 DATA 7,1,7,1,10,1,12,1,14,.
5,12,1.5,10,1,12,1,7,1,7,1,5,.5,
4,1.5,0,2
1060 FOR a=1 TO 28
1070 READ w: READ x
1080 BEEP x/10,w
1090 NEXT a
1095 IF h<s THEN LET h=s
1100 PRINT AT 11,9; PAPER 4; INK
 0;"DEN REKORD ";h;AT 12,9;"HAEL
T: ";e$
1105 RETURN
1110 LET v=v+RND
1120 GO TO 77
1150 FOR d=24 TO 0 STEP -1
1155 BEEP .01,d
1160 NEXT d
1165 BEEP .1,36
1170 LET s=s+5
1180 LET l$=","
1200 LET ba=INT (RND*20)+1
1210 LET rl=INT (RND*29)+1
1220 LET b$=SCREEN$ (ba,rl)
1230 IF b$="." OR b$="X" OR (b$=
"," AND s2<112) OR (ba>8 AND ba<
13) THEN GO TO 1200
1240 PRINT AT ba,rl; FLASH 1; IN
K 5; PAPER 0;"G"
1245 IF s>=112 THEN LET s2=s2+1
1250 RETURN
```

Ausschnitte des Spiels:



Variablen für den REBENKLAU:

h	höchster Punktestand bis jetzt
e$	Name desjenigen, der bis jetzt die meisten Punkte hat
a	Kontrollvariable für die verschiedenen FOR/NEXT-Schleifen
b	Kontrollvariable für eine FOR/NEXT-Schleife
c	holt die vom Benutzer definierten grafischen Daten
s	gegenwärtiger Punktestand
v	Anzahl der Reihen, die ein Winzer überspringen kann
z	INK-Farbe für die Mauern
y	INK-Farbe für die Weintrauben
f	Schalter für die Bildschirmwiedergaberoutine
u	y-Koordinate für Jacques

p	x-Koordinate für Jacques
du	Auf- oder Abwärtsbewegung für Jacques (kann −1, 0 oder +1 sein)
lj	Reihe der Weintrauben, in der sich Jacques befindet (1−4)
dp	Rechts- oder Linksschwenk von Jacques (siehe du)
lf	Reihe der Weintrauben, in der sich der Winzer befindet (siehe lj)
a	y-Koordinate des Winzers
g$	Zeichen, das den Winzer verschwinden läßt
t	x-Koordinate des Winzers
l$	Zeichen, das Jacques verschwinden läßt
da	Auf- oder Abwärtsbewegung des Winzers (siehe du)
dt	Rechts- oder Linksschwenk des Winzers (siehe dp)
q	Anzahl der Reihen, die der Winzer überspringt
c	Zeichen, um Jacques zu zeichnen
ba	y-Koordinate der „Supertraube“
rl	x-Koordinate der „Supertraube“
m	Schalter für den Fall, daß eine Weintraube gegessen wurde
i	Kontrollvariable in einer FOR/NEXT-Schleife
j	liest die erlaubten Richtungen, um Jacques von Hand zu bewegen
j$, k$	liest das erlaubte Zeichen in dieser Routine
u1, p1	Zwischenwerte für die Koordinaten von Jacques, während die Routine „Jacques-von-Hand-bewegen“ prüft, ob er richtig weiterkommt
n	ändert die Variable der Reihe, in der sich Jacques befindet
o	Richtung, in die der Winzer gehen muß, um auf Jacques zu treffen
b$	liest Zeichen, die das Verschwinden von Jacques andeuten sollen
r	liest die Tonhöhe der Klänge, bei denen Jacques verschwindet
s2	Anzahl der Weintrauben, die bisher auf diesem Spielfeld gegessen wurden
w	liest die einzelnen Noten der Eröffnungsmelodie
x	liest die Tonlängen der Eröffnungsmelodie
i$	Tastendruck, der die Routine „Jacques-von-Hand-bewegen“ abschaltet

Variablen, mit denen Daten eingelesen werden, ändern laufend ihren Wert. Sie sind jeweils eingebettet in eine FOR/NEXT-Schleife.

Gedicht

Die ersten beiden Spiele in diesem Kapitel hatten mit grafischen Zeichen zu tun. Die Aufgabe des Spielers bestand darin, die richtige Taste zu tippen, um die entsprechende Bewegung durchzuführen.

Die aktive Beteiligung des Spielers beim nächsten Spiel ist gleich Null. Er

kann sich nur zurücklehnen und die Ergebnisse bewundern. Geben Sie zunächst das Programm so ein, wie es ist, und schauen Sie, was dann passiert. Erst dann beginnt nämlich der interessante Teil des Spiels: Sie können die Data-Zeilen nach Ihren eigenen Wünschen abändern bzw. weitere Wörter hinzufügen. Was dabei herauskommt, sieht dann so aus:

```
DER SCHATTEN STAND TRAURIG IM TA
L,
RUFEND NACH EINEM AUSWEG,
UM DEN MOND ZU STEHLEN...
AUCH DIE INHALTE BESTAETIGEN ES


DAS ECHO STAND LEISE IM TAL,
RUFEND NACH EINEM AUSWEG,
UM DEN SCHMERZ ZU TEILEN...
AUCH DIE EINSICHTEN ENTSCHEIDEN


DER PROPHET SCHRIE LANGSAM IM DU
NKLEN WALD,
RUFEND NACH EINEM LICHT,
UM DEN MOND ZU BERUEHREN...
JEDOCH DIE ERFOLGE BEFRIEDIGEN


DER METZGER WARTETE VERRUECKT IM
 TAL,
STREBEND NACH EINEM WEG,
UM DEN SPASS ZU BEWEGEN...
AUCH DIE FREUDEN ENTSCHEIDEN
```

Und so sieht das Programm aus:

```
   10 REM GEDICHT
   20 DIM a$(13,12): DIM b(13)
   30 RANDOMIZE : GO TO 90
   40 FOR m=1 TO 11
   50 RESTORE 190+10*m
   60 FOR g=1 TO RND*10+1: READ b
$: NEXT g: LET a$(m)=b$
   70 LET b(m)=LEN b$: NEXT m
   80 RETURN
   90 POKE 23692,0
  100 FOR h=1 TO 10
  110 GO SUB 40
  120 PRINT      '"";a$(1)( TO b(1
));" ";a$(2)( TO b(2));" ";a$(3)
( TO b(3));" ";a$(4)( TO b(4));"
,"
  130 PRINT a$(5)( TO b(5));" nac
h einem "   ;a$(6)( TO b(6)): PRI
NT "um ";a$(7)( TO b(7));" zu ";
a$(8)( TO b(8));"..."
  140 PRINT a$(9)( TO b(9));" ";a
$(10)( TO b(10));" "     ;a$(11)(
TO b (11))
  160 PAUSE 200
  170 INK RND*6
  180 NEXT h
  190 STOP
  200 DATA "Der Mann","Der Junge"
,"Die Frau","Das Kind", Der Scha
tten","Das Echo","Das Maedchen",
"Das Pferd","Der Prophet","Der M
etzger"
  210 DATA "sang","wartete","sagt
e","heulte","schrie","aeusserte
sich","betete","stand","fiel","s
tolperte"
  220 DATA "traurig","gluecklich"
,"langsam","schwach","verrueckt"
,"demuetig","laut",leise","ruhig
","schnell"
  230 DATA "in der Dunkelheit","i
m Torweg","in der Einfahrt","am
Morgen","am Abend","im Vorhof","
im Bogengang","im dunklen Wald",
"im Tal","im gruenen Tann"
  240 DATA "schauend","suchend","
sich sehnend","tastend","verlang
end","strebend","fragend","rufen
d","laechelnd","handelnd"
  250 DATA "Weg","Pferd","Schritt
","Zeichen","Wunsch","Beduerfnis
","Licht","Tor","Ausweg","erloes
endes Ende"
  260 DATA "den Mond","die See","
die Nacht","den Schmerz","die Li
ebe","den Hass","die Sonne","die
 Furcht","den Spass","das Licht"
,"den Tag"
  270 DATA "erreichen","beruehren
","bewegen","stehlen","teilen","
wenden","drehen","betrachten","s
tossen","tadeln"
  280 DATA "nur","bloss","lediglich
 ","ach,"auch","doch","denn","
und","ja","jedoch"
  290 DATA "die Erfolge","die Erg
```

```
ebnisse","die Einsichten","die P
unkte","die Bedeutungen","die In
halte","die Freuden","die Beteil
igten","die Anwesenden"
  300 DATA "zahlen","gelten","ent
scheiden","befriedigen","bestaet
igen es","sagen es","druecken es
 aus","treffen","hoffen","brauch
en wir"
```

Es ist ziemlich leicht, Gedichtprogramme für einen Computer zu schreiben, wenn Sie folgendermaßen vorgehen: Sie müssen zunächst ein eigenes Gedicht schreiben. Danach müssen Sie den grammatikalischen Zusammenhang herausarbeiten (Subjekt, Prädikat, Objekt etc.) und zu jedem Typ zehn Wörter angeben. Der Computer kann dann nach Belieben der Reihe nach zu jedem Typ ein Wort auswählen.

Das vorliegende Programm wurde auf der Grundlage des folgenden, etwas holprigen Verses entwickelt:

Der Junge stand fragend am Ufer,
ausschauend nach einem Nachen,
um die Insel zu erreichen;
doch die Wellen schweigen.

Sobald Sie zu jedem Typ in den DATA-Zeichen Ihre Wörter gefunden haben, ist der Aufbau des eigentlichen Programms nicht mehr schwer. Zeile 20 legt zwei Felder an, eins zur Aufnahme der ausgewählten Wörter und eins zur Ermittlung der jeweiligen Wortlänge. Da insgesamt elf Wörter in jedem Gedicht verwendet werden (ohne die Füllwörter), wird in Zeile 40 eine Schleife von 1 bis 11 benötigt. Das Wort RESTORE bedeutet, daß der Computer das Lesen der DATA-Eintragungen mit der ersten DATA-Zeile beginnen soll (hier Zeile 200). Der Spectrum kennt noch eine besondere Variante dieses Befehls: Er kann nämlich genau die gewünschten DATA-Zeilen angeben (hier 190+10*M). Damit ist sichergestellt, daß er in Zeile 60 zum Lesen die jeweils benötigte Wortgruppe im Zugriff hat. Innerhalb der Wortgruppe wird bis zu einer zufällig gewählten Position gelesen. In Zeile 70 wird dann für die Eintragung in Feld B die Wortlänge ermittelt.

Der eigentliche Programmanfang befindet sich in Zeile 90. Dort wird der Speicherplatz 23692 auf Null gesetzt (POKE 23692,0). Hierdurch wird erreicht, daß der Bildschirm, wenn er voll ist, automatisch nach oben rollt. Die dauernde Anfrage „scroll?“ entfällt dann. Zeile 100 legt fest, wieviel Verse produziert werden. Im Beispiel werden zehn Verse erstellt. Diese Zahl kann nach Belieben verändert werden.

Das Unterprogramm ab Zeile 40 haben wir bereits besprochen. Es wird in Zeile 110 aufgerufen, um die Wörter auszuwählen, im Feld a$ einzutragen und deren Länge in Feld b.

Zeilen 120 bis 150 drucken das Gedicht. Die etwas umständliche Formulierung mit (TO b(1)) wird benötigt, um die anhängenden Leerzeichen der gespeicherten Wörter zu vermeiden. Ohne dies würde der Text ziemlich merkwürdig, mehr wie eine Tabelle aussehen.

Nach jedem Gedicht gibt es eine kurze Pause, die Farbe der Schrift wird geändert (Zeile 170), und der Computer fährt fort mit dem nächsten Vers.

Wie schon eingangs erwähnt: Richtig Spaß macht dieses Programm erst, wenn Sie selbst sich geeignete Wörter aussuchen. Wenn Sie diese Aufgabe mit Erfolg bestanden haben, werden Sie wahrscheinlich den Wunsch haben, ein Gedicht von Grund auf selbst zu entwickeln. Sie werden sehen: das ist gar nicht so schwer.

Meteoriten

In diesem Spiel fliegen Sie als Beobachtungsstation im Weltall und geraten plötzlich in einen Meteoritensturm. Sie müssen versuchen, den Meteoriten so lange wie möglich auszuweichen. Glücklicherweise finden Sie mitten im Sturm einige Treibstoffdepots und können Ihren Vorrat auffüllen. Dies geschieht dadurch, daß Sie ganz einfach in das Depot hineinfliegen (die moderne Technik macht's möglich).

Sie erinnern sich: Im GEDICHT-Programm (Zeile 90) haben wir die Speicherstelle auf Null gesetzt, um den Bildschirm automatisch durchlaufen zu lassen. Wir wollen das gleiche in diesem Spiel machen. Tatsächlich werden wir etwa in der Bildschirmmitte positioniert, und das Universum „rollt" wie ein Film an uns vorbei. Die geschickte Methode hierzu ist die, das Zeichen für einen neue Zeile (NEWLINE) am unteren Bildschirmrand anzugeben. Dadurch wird der Bildschirm eine Zeile nach oben geschoben, um für die neue Zeile Platz zu machen. Das benötigte Zeichen hat den Wert 13 (CHR$ 13).

Verwenden Sie die grafischen Buchstaben, die in den REM-Angaben stellvertretend für die symbolischen stehen. Obwohl Sie die Symbole nicht sofort nach dem Durchlaufen der Zeilen 5–70 sehen, sind doch die grafischen Buchstaben in frei definierte Zeichen des Anwenders umgeändert worden. Bei einem Probelauf dieser Zeilen sehen Sie jedes Mal einen Meteoriten auf dem Bildschirm auftauchen, sobald Sie das grafische Zeichen b eingeben.

Die Treibstoffanzeige ist in einem String gespeichert. Bei jedem Tastendruck wird diese Anzeige um eine Punktbreite reduziert und das Raumschiff steuert nach rechts (wenn Sie keine Taste drücken, haben Sie auto-

matisch eine Kursabweichung nach links). Bei zu hoher Geschwindigkeit verbrauchen Sie eimerweise Treibstoff. Ihr Raumschiff steuert dann einen x-beliebigen Punkt an und könnte dabei auf einen Meteoriten treffen. Greifen Sie also nur im Notfall zu diesem Mittel. Wenn Ihr Treibstoff knapp wird, sollten Sie das Raumschiff treiben lassen (das verbraucht keinen Treibstoff). Aber Vorsicht! Wenn Sie sich am linken Rand des Bildschirms befinden, verlieren Sie Punkte, statt welche hinzuzugewinnen. Sie können auftanken, wenn Sie eine blaue Treibstoffpumpe treffen, erkennbar an der blauen Farbe. Ihre Station explodiert, sobald der Treibstoff ausgegangen ist, bzw. wenn die Station von einem Meteoriten getroffen wird. Die Explosion kommt auf die gleiche Art und Weise wie im Programm NÄCHTLICHER ÜBERFALL zustande. Nach jeweils 500 Punkten werden die Meteoriten größer, und auch die Explosion wird dann stärker ausfallen. Sie weichen aus durch Drücken der Leertaste. Alle anderen Tasten bewirken eine Bewegung nach rechts.

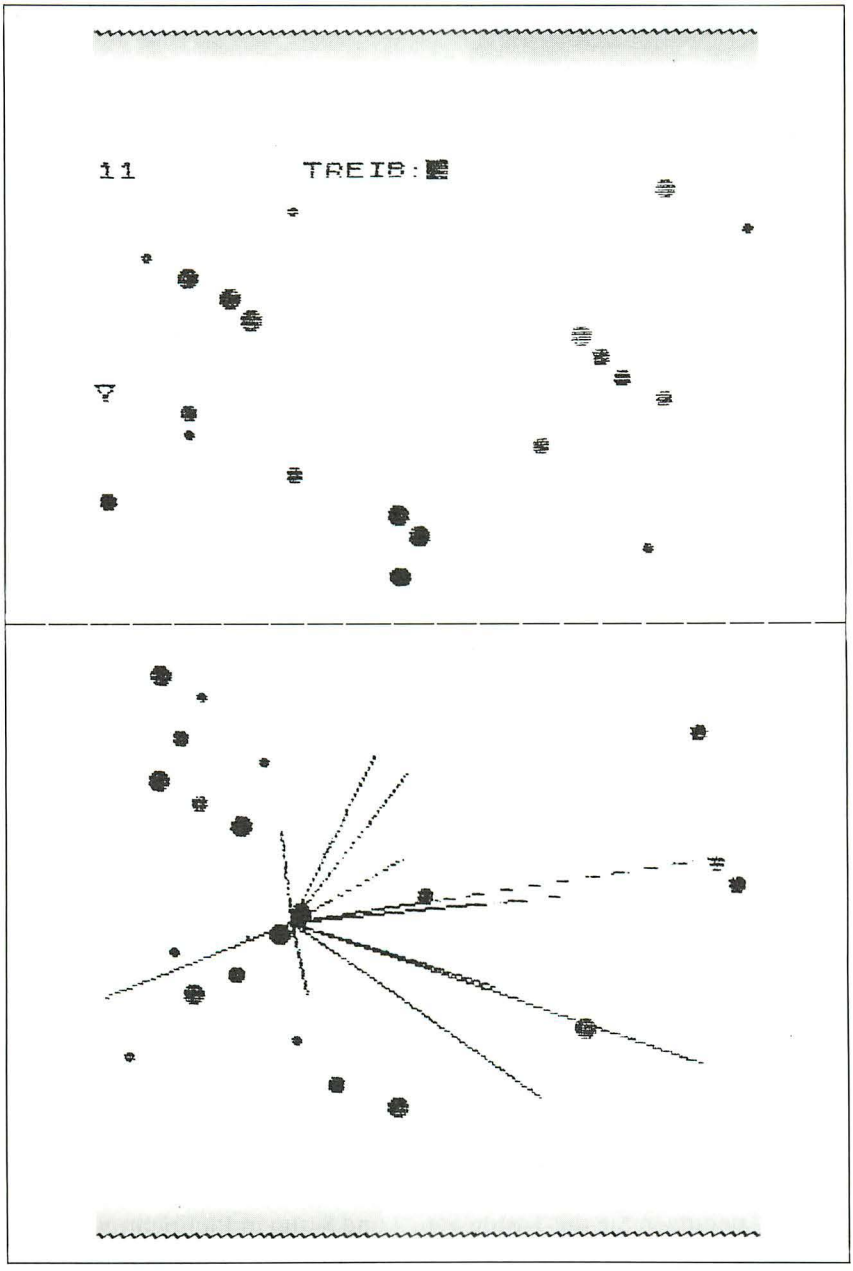
Vielleicht möchten Sie das Programm noch etwas verbessern und Gefechtsmöglichkeiten oder noch zusätzlich feindliche Raumschiffe in den Meteoritensturm einbauen, die dann von Zeit zu Zeit auftauchen.

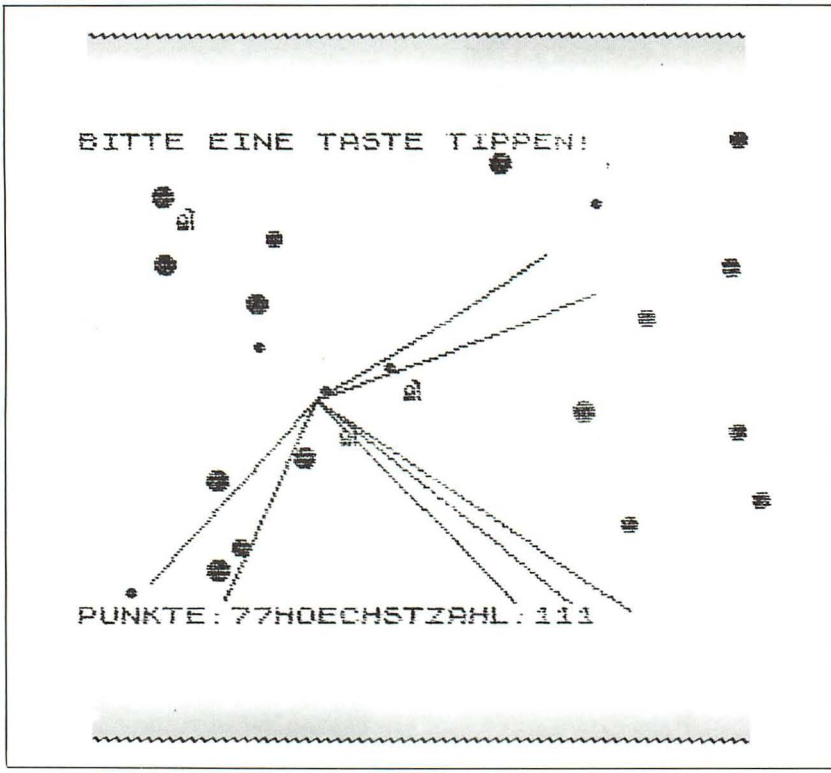
Programmliste METEORITEN:

```
  5 LET h=0
 10 LET u=11
 20 LET p=16
 22 LET s=0
 25 IF h>0 THEN GO TO 80
 30 DATA "a",255,66,66,36,24,24
,24,0,"b",60,126,255,255,255,255
,126,60
 32 DATA "c",0,60,126,126,126,1
26,60,0,"d",0,0,24,60,60,24,0,0
 35 DATA "m",12,6,50,74,74,122,
122,122
 35 FOR b=1 TO 5
 37 READ a$
 40 FOR a=0 TO 7
 50 READ c: POKE USR a$+a,c
 60 NEXT a
 70 NEXT b
 72 DATA 255,127,63,31,15,7,3,1
 74 FOR a=0 TO 7: READ c
 76 FOR b=0 TO 7: POKE USR CHR$
(a+101)+b,c
 78 NEXT b: NEXT a
 80 BORDER 0: PAPER 0: INK 6: C
LS
 82 LET a$="                "
 84 LET sl=16
 86 LET sa=148
 90 REM Grafik-A = Rakete
100 REM Grafik-B = Meteorit usr.
```

```
 110 REM Grafik-C = Meteorit
 120 REM Grafik-D = Meteorit kl.
 125 REM Grafik-M = Treibstoff
 130 IF SCREEN$ (u,p)<>" " THEN
GO TO 200
 132 PRINT INK 4;AT u,p;"A"
 135 POKE 23692,255
 137 LET s=s+1: PRINT AT 0,0;s
 139 PRINT AT 0,10;"Treib:"; INK
 (sl>=25)*-1+6;TAB sl;a$
 140 FOR a=1 TO s/500+1: PRINT I
NK 2;AT 21,INT (RND*32);CHR$ (14
5+INT (RND*3)): NEXT a
 141 IF RND>.95 THEN PRINT INK 5
;AT 21,INT (RND*31);"M"
 142 IF INKEY$<>"" THEN LET sa=s
a+1: IF sa>=156 THEN LET sl=sl+1
: LET a$=a$(2 TO ): LET sa=148
 143 IF sl=31 AND sa=148 THEN GO
TO 200
 144 LET a$(1)=CHR$ sa
 147 IF sl>=25 THEN BEEP .007,sl
 150 PRINT AT u,p;" "
 155 IF INKEY$=" " THEN LET p=IN
T (RND*32) : IF LEN a$<=1 THEN G
O TO 200: LET a$=a$(2 TO ): LET
sl=sl+1
 160 PRINT AT 21,0;CHR$ 13
 170 LET p=p-(p>0)+(INKEY$<>"")*
2*(p<31)
 175 IF p=0 THEN LET s=s-2*(s>0)
: BEEP .01,s/20
 180 GO TO 130
 200 IF ATTR (u,p)=5 THEN LET a$
="                ": LET sl=16:PR
INT AT u,p;"*": FOR a=-sl TO 30
: BEEP .01,a: NEXT a: GO TO 132
 202 LET x=p*8: LET y=(21-u)*8
 205 INK 3
 207 FOR c=1 TO s/10
 210 PLOT x,y
 220 LET a=INT (RND*256): LET b=
INT (RND*158)
 230 DRAW a-x,b-y
 235 IF RND>.85 THEN BEEP .01,20
 240 NEXT c
 250 INK 6
 260 BEEP 1,20
 265 IF h<s THEN LET h=s
 270 PRINT AT 21,0;"PUNKTE:";s;"
HOECHSTZAHL:";h
 280 PRINT AT 0,0;"BITTE EINE TA
STE TIPPEN!"
 290 IF INKEY$="" THEN GO TO 290
 300 GO TO 10
```

Beispiellauf METEORITEN:





Schädelacker

Bei einem so grauenhaften Namen kann man eigentlich kaum erwarten, daß das Spiel Spaß macht. Es ist einfach zu spielen, aber die Gewinnchancen sind sehr gering.

Sie spielen in diesem Spiel eine kleine, menschenähnliche Kreatur (siehe Zeile 140), die von zehn Schädeln verfolgt wird. Die Schädel leuchten ständig auf, wissen, wo ihr Opfer zu finden ist und verfolgen es fast dauernd. Weil die Schädel aber ziemlich dumm sind, kann man sie leicht vernichten. Man lockt sie einfach in die farbigen Rechtecke, die über den Bildschirm verstreut sind. Das gelingt jedoch nur, wenn man ein Rechteck zwischen sich und den Schädel bringt. Dann laufen sie blindlings in das Rechteck hinein (weil sie nur Augen für ihr Opfer haben) und lösen sich mit einem komischen „blupp" auf. Die Rechtecke sollen hier Gräber andeuten.

Im Spiel benutzen Sie die Tasten 5, 6, 7 und 8, um in Pfeilrichtung zu gehen. Sie können nicht durch die farbigen Rechtecke hindurchgehen. Die

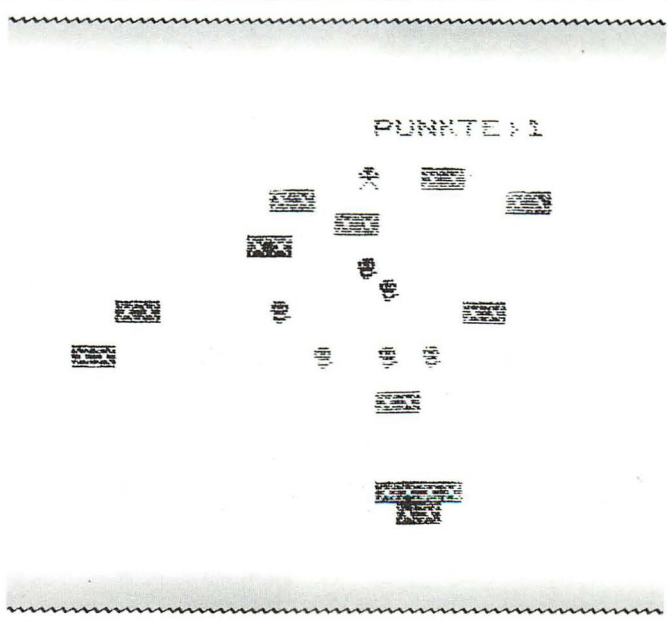
Berührung mit diesen Stellen ist ungefährlich. Nur ein Schädel kann Ihnen gefährlich werden. Das Spiel ist zu Ende, wenn Sie von einem Schädel erwischt werden, oder wenn Sie alle Schädel in die Gräber gelockt haben.

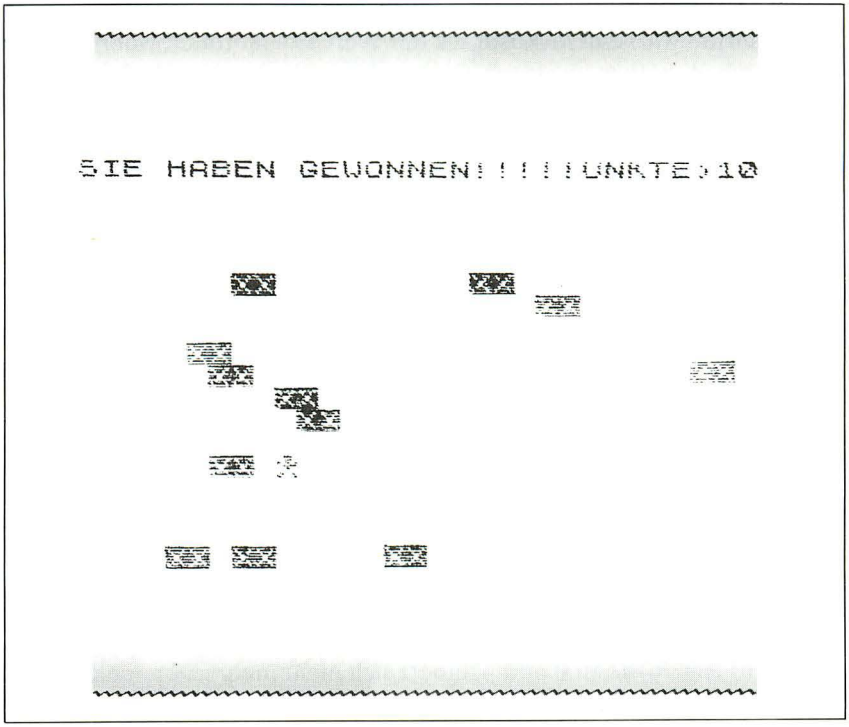
Sie erinnern sich sicher, wie wir in den anderen Spielen dieses Kapitels grafische Zeichen definiert haben. Es ist sicherlich etwas umständlich, für jedes Zeichen eine eigene Schleife zum Einlesen zu schreiben. Man kann auch mehrere Zeichen gleichzeitig in einer Schleife definieren. Aber das ist mitunter etwas unübersichtlich. In den Zeilen 400 bis 430 werden sowohl die Kreatur als auch die Schädel festgelegt. Die DATA-Angabe in Zeile 440 enthält die grafische Information für beide Zeichen in abwechselnder Reihenfolge. Mit A wird die Figur und mit B ein Schädel gezeichnet. In Zeile 70 wird daher ein Grafik-A verwendet.

```
  10 REM SCHAEDELACKER
  20 GO SUB 310: REM VARIABLE
  30 FOR b=1 TO 10
  40 IF a(b,1)=1 THEN GO SUB 220
  50 PRINT AT c(b,2),c(b,3);" "
  60 IF a(b,1)<>1 THEN GO TO 180
  70 PRINT AT a(b,2),a(b,3); INK
b/2; FLASH 1;"b"
  80 LET c(b,2)=a(b,2): LET c(b,
3)=a(b,3)
  90 LET n=n-(INKEY$="5" AND n>0
)+(INKEY$="8" AND n<31)
 100 IF a(b,2)>m THEN LET a(b,2)
=a(b,2)-1
 110 LET m=m+(INKEY$="6" AND m<1
9)-(INKEY$="7" AND m>0)
 120 IF a(b,2)<m THEN LET a(b,2)
=a(b,2)+1
 130 IF SCREEN$ (m,n)="X" THEN L
ET m=x: LET n=y
 140 PRINT AT x,y;" ";AT m,n;"a"
 150 IF a(b,3)<n THEN IF a(b,3)<
28 THEN LET a(b,3)=a(b,3)+INT (4
*RND)
 160 LET x=m: LET y=n
 170 IF a(b,3)>n THEN IF a(b,3)>
3 THEN LET a(b,3)=a(b,3)-INT (4*
RND)
 180 NEXT b
 190 IF punkt<10 THEN GO TO 30
 200 PRINT AT 2,0; FLASH 1; BRIG
HT 1; INK 2; PAPER 6;"SIE HABEN
GEWONNEN!!!!!"
 210 BEEP .008,60-120*RND: GO TO
210
 220 REM ■■ SCHAEDEL PRUEFEN ■■
 230 IF SCREEN$ (a(b,2),a(b,3))=
"X" THEN LET a(b,1)=0: FOR h=1 T
O 10: BEEP .01,5*h: BEEP .01,50-
5*h: NEXT h: LET punkt=punkt +1:
PRINT AT 2,22; FLASH 1; BRIGHT 1
; INK 4;"PUNKTE>";punkt
 240 IF a(b,2)=m AND a(b,3)=n TH
EN PRINT AT c(b,2),a(b,3);" ": G
O TO 260
 250 RETURN
 260 REM ■■ SPIELENDE ■■
 270 PRINT AT a(b,2),a(b,3); FLA
SH 1; BRIGHT 1; INK 2;"a"
 280 BEEP .01,RND*20+40
```

```
  290 GO TO 280
  300 STOP
  310 REM ■■ VARIABLEN ZUWEISEN ■
■
  320 LET punkt=0
  330 DIM a(10,3): DIM c(10,3)
  340 FOR b=1 TO 10
  350 LET a(b,1)=1
  360 LET a(b,2)=10+INT (RND*9)
  370 LET a(b,3)=10+INT (RND*19)
  380 LET c(b,2)=a(b,2): LET c(b,
3)=a(b,3)
  390 NEXT b
  400 FOR j=0 TO 7
  410 READ a: READ b
  420 POKE USR "a"+j,a: POKE USR"
b"+j,b
  430 NEXT j
  440 DATA 28,60,28,126,73,90,127
,126,8,60,20,35,34,60,65,24
  450 LET m=INT (RND*3)+9
  460 LET n=30-INT (RND*30)
  470 LET x=m: LET y=n
  480 PAPER 7: BORDER 7: CLS
  490 INVERSE 1
  500 FOR g=1 TO 12
  510 PRINT INK g/2;AT RND*16+4,R
ND*26+4;"XX"
  520 NEXT g
  530 INVERSE 0
  540 RETURN
```

Ein Ausschnitt aus dem SCHÄDELACKER:





Ausbruch

Ziel dieses Spiels ist es, mit einem Ball möglichst viele Steine aus einer Mauer herauszuschlagen, die am oberen Bildschirmrand zu sehen ist. Ein Schläger am unteren Rand kann nach links und rechts bewegt werden und muß versuchen, den zurückkommenden Ball wieder gegen die Mauer zu schlagen. Wenn der Ball nicht getroffen wird, ist das Spiel zu Ende.

Das Spiel besteht aus fünf Teilen:

1. Zeichnen der Ziegelsteinmauer
2. Bewegen des Schlägers bei Bedarf
3. Abprallen des Balls am Bildschirmrand
4. Zählen der herausgeschlagenen Ziegelsteine
5. Spielabbruch, wenn der Schläger den Ball verfehlt.

Teil eins ist leicht und bedarf keiner Erläuterung. Teil zwei prüft, welche Taste gedrückt wurde, ändert entsprechend die Position des Schlägers und verhindert, daß der Schläger über den Spielfeldrand hinausgeht.

All das geschieht in einer einzigen Zeile. Denn entsprechend der Logik des Spectrum wird einem Ausdruck der Wert 1 zugeordnet, wenn er wahr ist, und der Wert 0, wenn der Ausdruck falsch ist. Der Ausdruck INKEY$ = „5" hat also den Wert 1, wenn die Taste 5 gedrückt wurde, und sonst den Wert 0.

Die Variable P dient als x-Koordinate des Schlägers. Sie wird im folgenden Ausdruck verwendet:

```
LET p=p + ((INKEY$ = "8") * (p<26) - (INKEY$ = "5") * (p>0)) * 2
```

Dieser Ausdruck ist nicht ganz so kompliziert, wie er aussieht. Als erstes fällt uns die Fomulierung (INKEY$ = "8") sowie (INKEY$ = "5") auf. Wenn keine Taste gedrückt wurde oder eine von 5 und 8 verschiedene Taste, haben beide Ausdrücke den Wert 0, d. h. egal was mit 0 multipliziert wird, es kommt immer 0 heraus; p verändert nicht seinen Wert. Jetzt wollen wir uns den Wert der beiden anderen Ausdrücke näher ansehen: (p<26) und (p>0). Diese Klammerausdrücke haben entweder den Wert 1 oder den Wert 0. Ist z. B. p=30, so wird der erste Ausdruck (p<26) wertmäßig 0, weil p eben nicht kleiner als 26 ist. Dagegen erhält (p>0) den Wert 1, denn 30 ist größer als 0. Wenn jetzt die Taste 8 getippt wird, erhält (INKEY$ = "8") den Wert 1 und wird mit (p<26)*2 multipliziert. Es wird also so lange p=p+2 gebildet, bis p den Wert 26 erreicht hat. Danach wird (p<26)*2 zu 0 und p ändert sich nicht mehr: Der Schläger hat den rechten Bildschirmrand erreicht. Genauso geht dies, wenn mit der Taste 5 der Schläger nach links bewegt werden soll. Der linke Rand ist erreicht, sobald p=0 ist. (p>0) ergibt 0 und es wird nichts mehr abgezogen.

Die eingeklammerten Vergleichsausdrücke können beliebig erweitert werden, z. B. kann man schreiben: (a+b=c). Daraus ergibt sich: (6+7=13) hat den Wert 1 (d. h. wahr) und (12+15=26) hat den Wert 0 (d. h. falsch).

Teil drei läßt den Ball von der Mauer abprallen, indem die Richtungsvariable des Balls umgedreht wird. Bei jedem Aufprall wird einfach die Richtungsvariable mit (−1) multipliziert: War sie vorher positiv, so wird sie jetzt negativ und umgekehrt. Bei der Ballbewegung wird dann jeweils eine 1 addiert (nach rechts) oder subtrahiert (nach links) bis zum nächsten Aufprall auf eine Wand. Beim Aufprall wird außerdem ein Ton erzeugt, um den Aufprall akustisch zu untermalen.

Teil 4 benutzt die ATTR-Funktion, um den Zusammenprall des Balls mit einem Ziegelstein festzustellen. Da alle Ziegelsteine gelb sind, gibt es jedesmal einen Punkt, wenn der Ball auf einen gelben Fleck trifft.

Teil 5 arbeitet mit SCREEN$. Der Schläger besteht aus großen X. Wenn der Ball am unteren Rand des Bildschirms angekommen ist, wird geprüft, ob dort ein X steht. Falls nicht, ist das Spiel zu Ende.

Wenn der Ball den Schläger trifft, prallt er unter einem zufälligen Winkel ab (niemals senkrecht). Wenn der Ball fast senkrecht fliegt, kann man ihn relativ leicht treffen. Ist die Flugbahn jedoch diagonal, ist der Aufschlagpunkt sehr schwer zu bestimmen. Zudem entsteht der Eindruck, daß der Ball wesentlich schneller wird.

Falls Sie möchten, können Sie das Programm noch ändern. Beispielsweise könnte der Ball gelegentlich bei einzelnen Ziegelsteinen zurückprallen, statt die ganze Mauer zu durchschlagen. Ferner fehlt noch ein Zähler für den Punktehöchststand. Ihn einzubauen dürfte sicherlich eine gute Übung sein. Schließlich müßten sich selbstdefinierte Grafik-Zeichen sehr gut machen. Dann muß jedoch die SCREEN$-Abfrage in eine ATTR-Funktion umgebaut werden (siehe NÄCHTLICHER ÜBERFALL). Ein Grafik-Ball dürfte nicht sehr schwierig sein. Die Ziegelsteine können nach eigenen Vorstellungen entwickelt werden. Wichtig ist lediglich die Farbe gelb für die ATTR-Funktion. Die Steine müssen nicht mal gleich aussehen. Sie können die Form nach Belieben laufend ändern.

Denken Sie daran: Die Aufgabe des Spiels ist es, alle Steine von oben herunterzuschlagen. Der Ball wird mit einem Schläger geschlagen, der am unteren Bildschirmrand mit den Tasten 5 und 8 nach links und rechts bewegt wird. Das Spiel endet, wenn der Schläger den Ball verfehlt. Für jeden heruntergeschlagenen Ziegelstein gibt es einen Punkt.

Programmliste: AUSBRUCH

```
  5 REM AUSBRUCH
 10 BORDER 0: PAPER 0: INK0
 20 CLS
 25 INK 6
 30 FOR a=1 TO 6
 40 FOR b=0 TO 31
 50 PRINT "■";
 60 NEXT b
 70 NEXT a
 80 LET u=21
 90 LET p=14
140 INK 4
150 LET a=u
155 LET s=0
160 LET t=p+1
162 LET dt=(INT (RND*5)+1)/(INT
(RND*5)+1)
165 LET da=-1
170 PRINT AT u,p; INVERSE 1,"XX
XXX"
172 PRINT AT 11,0;s
173 IF INKEY$="z" THEN COPY
175 PRINT AT z,t;" "
180 LET a=a+da
190 LET t=t+dt
195 IF a=0 OR a=21 OR t<=0 OR t
>=31 THEN GO TO 240
```

```
 197 IF ATTR (a,t)=6 THEN LET s=
s+1
 200 PRINT AT a,t; INVERSE 1; IN
K 1;"O"
 210 PRINT AT u,p;"     "
 220 LET p=p+((INKEY$="8")*(p<26
)-(INKEY$="5")*(p>0))*2
 225 LET p=p+(INKEY$="8")*(p=26)
-(INKEY$="5")*(p=25)
 230 GO TO 170
 240 IF a=21 AND SCREEN$ (a,t)="
X" THEN GO TO 162
 250 IF a=21 THEN GO TO 310
 260 IF t<=0 THEN LET t=ABS t: L
ET dt=-dt
 270 IF t>=31 THEN LET t=31:LET
dt=-dt
 280 BEEP .1,-10
 290 IF a=0 THEN LET da=1
 300 GO TO 200
 310 PRINT AT a,t; INK 2;" "
 320 BEEP 1,15
 330 IF INKEY$<>"" THEN GO TO 33
0
 340 IF INKEY$="" THEN GO TO 340
 350 RUN
```

Spielfieber

Dieses Programm ist eine Variation des „einarmigen Banditen". Hier kommen die selbstentworfenen grafischen Zeichen in Viererblöcken wirkungsvoll zum Einsatz. Hergestellt wird ein Geldspielautomat, der die hervorragenden Eigenschaften des Spectrum voll zur Geltung bringt. Die Programmliste läßt nicht im entferntesten ahnen, wie faszinierend dieses Spiel auf einem Farbfernseher wirkt.

Das Konzept des Spiels läßt sich mit wenigen Worten umreißen: Wir starten mit einem Geldeinsatz von 50 DM. Der Automat hat vier Sichtfenster, hinter denen die Glücksräder rotieren. Die Räder zeigen in zufälliger Reihenfolge eines der drei Symbole GLOCKE, KIRSCHE oder APFEL. Von jeder Sorte sind gleichviel auf jedem Rad. Wenn die Walzen anhalten, sehen wir hinter jedem Sichtfenster je ein Symbol. Wenn drei von ihnen übereinstimmen, gewinnen wir 35 DM. Wenn alle vier gleich sind, gewinnen wir 100 DM.

Das Spiel läuft so lange, bis wir entweder pleite sind oder den Automaten sprengen (bei einem Gewinn von 500 DM). Jedes Spiel kostet 5 DM und wird bei Beginn vom Guthaben abgezogen (oben rechts auf dem Bildschirm).

Von Zeit zu Zeit wird ein STOP-Mechanismus wirksam. Das Wort STOP

beginnt zu blinken, und sie können dann irgendeins der Sichtfenster für die nächste Runde anhalten. Hierzu geben Sie die Nummer des Fensters ein (von links nach rechts gezählt), anschließend ENTER. Nach Eingabe von ENTER stoppt die angegebene Walze. Wenn Sie keine Nummer eingegeben haben, laufen die Walzen weiter.

Es gibt noch eine weitere Funktion: Die Glocke hat den höchsten Wert. Sie sollten daher versuchen, bei STOP das Fenster anzuhalten, welches eine Glocke zeigt. Zwei Glocken nebeneinander geben 15 DM, drei Glocken geben 50 DM (statt sonst 35). Der Bonus wird nicht extra angezeigt, sondern sofort aufsummiert.

Dieses Programm ist lediglich als Konzept gedacht, in das Sie Ihre eigenen Ideen und Wünsche einbauen können: besondere Symbole, unterschiedliche Klänge und Melodien, ein Anstoßmechanismus, um nur einige der Möglichkeiten zu nennen. Lassen Sie Ihrer Phantasie freien Lauf!

Im Kapitel „Tips und Tricks zur Programmierung" werden Sie einige Vorschläge finden, um Programme zu überarbeiten, sowohl im Stadium der Planung als auch während der Programmierung. Ein wichtiger Tip ist der, zunächst die wichtigsten Ideen zu formulieren und für jede Idee eine eigenständige Routine zu schreiben. Diese Routinen können dann im Hauptteil des Programms nacheinander aufgerufen werden. Genauso wurde dieses Programm entwickelt.

Am Anfang des Programms werden drei Unterprogramme aufgerufen. Mit GO TO 40 springt das Programm zwei Zeilen zurück, um die zweite und dritte Routine zu wiederholen. In der ersten Routine (ab Zeile 720) werden die Startwerte der Variablen gesetzt und die Grafik-Zeichen gebildet. Die zweite Routine dreht ständig die Walzen. Diese Routine beginnt ab Zeile 290 und ruft in jeder Runde achtmal eine weitere Routine (ab Zeile 210) auf. Die letzte Unterroutine wird nur aufgerufen, wenn die berechnete Zufallszahl (in Zeile 50) größer als 45 ist. Sie bringt die STOP-Variante ins Spiel.

Das Programm wurde systematisch von oben nach unten („top down") geschrieben. Zunächst wurden die drei Hauptfunktionen – Startwerte, Drehen der Walzen und die STOP-Routine – festgelegt und als Unterprogramme aufgerufen. Zum Schluß wurde das Programm umnumeriert (RENUMBER). In der ursprünglichen Version begann die Startroutine bei 9000 und das Drehen der Walzen bei 8000, die Darstellung der Walzen bei 6000 und das Stoppen der Walzen bei 3000.

Das Programm ist durch die Sternchenreihe in den REM-Zeilen deutlich in Abschnitte unterteilt. So sehen wir recht gut, welcher Teil des Programms welche Aufgabe hat, und können ohne Schwierigkeiten Änderungen anbringen. Z. B. können wir den Teil 210–280 zur Darstellung der Walzen ändern, ohne dadurch die anderen Routinen zu beeinflussen.

Wenn Sie auf ähnliche Weise vorgehen und sich ein bißchen an die Tips und Tricks im letzten Kapitel anlehnen, werden Sie in der Lage sein, Programme zu entwickeln, die mit einem Minimum an Tests auskommen, die gut lesbar sind und die auch nach einer längeren Pause schnell wieder verstanden werden.

Schauen wir uns jetzt das Programm Abschnitt für Abschnitt an und betrachten wir, was die einzelnen Routinen machen. Wir haben die Zeilen 10–60 bereits besprochen. Diese sind für den Programmablauf zuständig. Wir gehen die Routinen in der Reihenfolge durch, in der sie aufgerufen werden. Denn so verstehen wir leichter, was im einzelnen gemacht wird.

Die erste Routine beginnt bei Zeile 720, die, wie man aus der REM-Zeile entnimmt, für die Variablen und die Grafik zuständig ist. Es ist immer gut, die Variablen erst am Schluß zuzuordnen. Das Programm läuft dann schneller, weil diese Zeilen nie wieder durchsucht werden müssen.

Die Variable GELD ist verständlicherweise für das Guthaben zuständig. Diese Variable beginnt mit dem Wert 50. Zeile 760 legt vier Felder an: Die ersten beiden sind Strings und enthalten später die selbstdefinierten Grafikzeichen sowie deren Farben; die nächsten beiden Felder enthalten die Zahlen für jede Drehung der Walzen (A) und Ihre Wünsche bezüglich STOP (Q).

Die Routinen von 770 bis 1030 legen die Grafik fest, die dann die DATA-Angaben von Zeile 1150 aufwärts aufruft. Die Zeilen 840 und 850 bilden in vier Teilen einen Apfel. Dazu gehört das Kontrollzeichen (CHR$ 16) für die Schrift und das Zeichen für rot (CHR$ 2). Damit wird sichergestellt, daß PRINT A$(1) beispielsweise die obere Hälfte des Apfels in rot zeichnet, ohne daß deswegen der Befehl PRINT INK 2;„AB“ notwendig wäre. Das erleichtert es auch, bei Bedarf das dazugehörige Symbol für die in Zeile 440 bereitgestellten Zufallszahlen abzurufen. Die Zeilen 930 und 940 fügen die Glockenteile in der Farbe gelb zusammen und Zeile 1020 bzw. 1030 das Bild der Kirsche in violettrot (genaugenommen sind es drei Kirschen).

In der Programmliste sehen Sie die bereits erstellten Symbolteile. An diesen Stellen müssen Sie die zugehörigen Grafik-Tasten tippen: A und B in Zeile 840, C und D in Zeile 850, E und F in Zeile 930, G und H in Zeile 940, J und K in Zeile 1020 sowie L und M in Zeile 1030.

Die Zeilen von 1040 bis 1130 setzen den Kasten zusammen. Die Walzen werden im Laufe des Spiels eingeblendet. Die ursprüngliche Form der Maschine und die Figuren sowie die Entscheidung darüber, wo verschiedene Wörter während des Spiels erscheinen sollen, wurden von einem kommerziellen Produkt namens „Print'n'Plotter Jotter“ abgeschaut. Dieser Apparat hat festgelegte Regeln, nach denen die Zähler zueinander in Beziehung stehen. Dies wurde beim ZX mit den PRINT AT-Befehlen

nachvollzogen. Dieser – oder auch ein ähnlicher – Apparat dürfte sicherlich auch für Sie eine gute Hilfe zur Entwicklung eigener Programme sein.

Nach dem RETURN dieses Unterprogramms gehen wir zur Zeile 300 und starten mit dem Drehen der Walzen. Zunächst wird in Zeile 320 bezahlt (DM 5). Als nächstes wird die Variable PREIS auf Null gesetzt. Zeile 350 fordert dazu auf, den Hebel herunterzudrücken, d. h. die ENTER-Taste zu tippen. Zeile 360 wartet, bis die Tastatur wieder frei ist, geht dann nach Zeile 370 und wartet auf einen neue Taste. Zeile 380 entfernt die Aufforderung „Drücke ENTER" vom Bildschirm. Ihr Guthaben (im Augenblick beträgt es 5 DM weniger) wird in Zeile 390 angegeben. Zeilen 400 bis 470 drehen die Walzen achtmal. Bei jedem Durchlauf durch die Z-Schleife wird jeweils die Routine 210 zur Darstellung der Walzen herangezogen.

Die B-Schleife innerhalb der Z-Schleife produziert vier beliebige Zahlen zwischen 1 und 4 und ordnet sie den Elementen 1 bis 4 der Tabelle A zu. Falls Sie die STOP-Routine angefordert haben, was durch den Wert im Q-Feld angezeigt wird, springt das Programm über die Zufallsroutine und geht direkt zu NEXT B. Zeile 430 imitiert das Automatengeräusch während des Rotierens. Dies erfolgt erst, wenn die STOP-Wahlmöglichkeit überprüft worden ist (siehe das Element des Feldes Q). Die Geräusche verändern sich dann je nachdem, ob eine oder mehrere Walzen angehalten wurden.

Zeile 480 setzt alle Elemente des Feldes Q auf 0. Das hat zur Folge, daß der STOP-Mechanismus gelöst wird. Die Zeilen 490 bis 510 überprüfen die verschiedenen Gewinnmöglichkeiten: 490 für den Haupttreffer (vier gleiche Symbole), 500 für drei gleiche und 510 für zwei oder mehr Glokken in einer Reihe.

Die Zeilen 520 bis 540 erzeugen einige Geräusche, bevor das Ergebnis des Spiels gezeigt wird. Wenn Sie gewinnen, erhöht sich Ihr Guthaben in Zeile 550 entsprechend. Zeile 570 nennt die Art des Gewinns (Q$) und Zeile 590 gibt den Gewinnbetrag (PREIS) auf dem Automaten an. Wenn Sie einen Bonus erzielen (zwei oder drei Glocken in einer Zeile), wird dies in Zeile 610 und 620 angezeigt. Zeile 630 erzeugt recht interessante, harfenähnliche Klänge am Ende des Spiels und verändert den Kontostand. Zeilen 650 bis 670 löschen die Gewinn- und Preisangaben. In den Zeilen 680 und 690 wird geprüft, ob Sie das Ziel des Spiels, die Bank zu sprengen, erreicht haben, oder ob Ihr Guthaben erschöpft ist. In beiden Fällen ist das Spiel zu Ende.

Die Routine zum Zeichnen der Walzen (ab Zeile 210) ist ziemlich klar. Hier werden die Grafik-Zeichen ausgegeben, die dem Feld A$ zugewiesen wurden.

Bleibt zuletzt noch die STOP-Routine. In Zeile 100 werden sieben reine Töne gespielt und danach das Wort STOP! ausgegeben. Dieser Teil enthält eine sehr nützliche Fehlerabfangroutine. Normalerweise würde man für eine Zahleneingabe den Befehl INPUT Q verwenden. Dies kann gelegentlich zum Programmabbruch führen, z. B. dann, wenn statt einer Zahl nur ENTER eingetippt wird. Um das zu vermeiden, wird zunächst die Eingabe einer Zahlenfolge mit INPUT Q$ verlangt. Die Zeile 130 prüft, ob nur die ENTER-Taste getippt wurde (GO TO 190). Wenn ein leerer String (also nur die ENTER-Taste) eingegeben wurde, springt das Programm zum Löschen der Fenster und beendet dann die Unterroutine mit RETURN. Anderenfalls wird in Zeile 140 geprüft, ob das erste Zeichen der Eingabe Q$(1) eine Ziffer ist. Die Umwandlung erfolgt mit der VAL-Funktion. Gibt der Spieler mehr als eine Ziffer ein, so werden diese einfach fallengelassen, da Q$(2) etc. unberücksichtigt bleiben. Die Zeile 150 notiert, welche Scheibe angehalten werden soll. Hierzu wird z. B. Q(1) = 1 gesetzt oder Q(4) = 4, je nachdem, welche Walze ausgewählt wurde. Zeile 170 nennt die Walze, die gestoppt werden soll und fragt dann, welche Walze als nächste zu stoppen ist (GO TO 120). Wenn alle gewünschten Walzen gestoppt worden sind und nur noch die ENTER-Taste ohne eine Ziffer getippt wurde, springt das Programm in Zeile 150, löscht die STOP-Meldungen und springt danach zurück in den Hauptteil zur Zeile 60. Von dort springt das Programm zur Zeile 40 und bildet so eine Endlosschleife.

Nachdem wir gesehen haben, wie das Programm aufgebaut ist, können wir jetzt ein Spiel wagen. Manches wird danach sicher noch klarer werden. Danach können Sie sich daran machen, z. B. zusätzliche Bilder auf die Walzen zu bringen, alternative Gewinnchancen anzubieten oder auch, wie wir eingangs gelesen haben, einen Anstoß-Mechanismus. Eine andere Variante wäre ein Automat mit vier Bildern auf drei Walzen oder ein Automat, auf dem auch das Bild davor und das Bild dahinter zu sehen ist. Das ergibt zusätzliche Gewinnmöglichkeiten durch die Diagonalen.

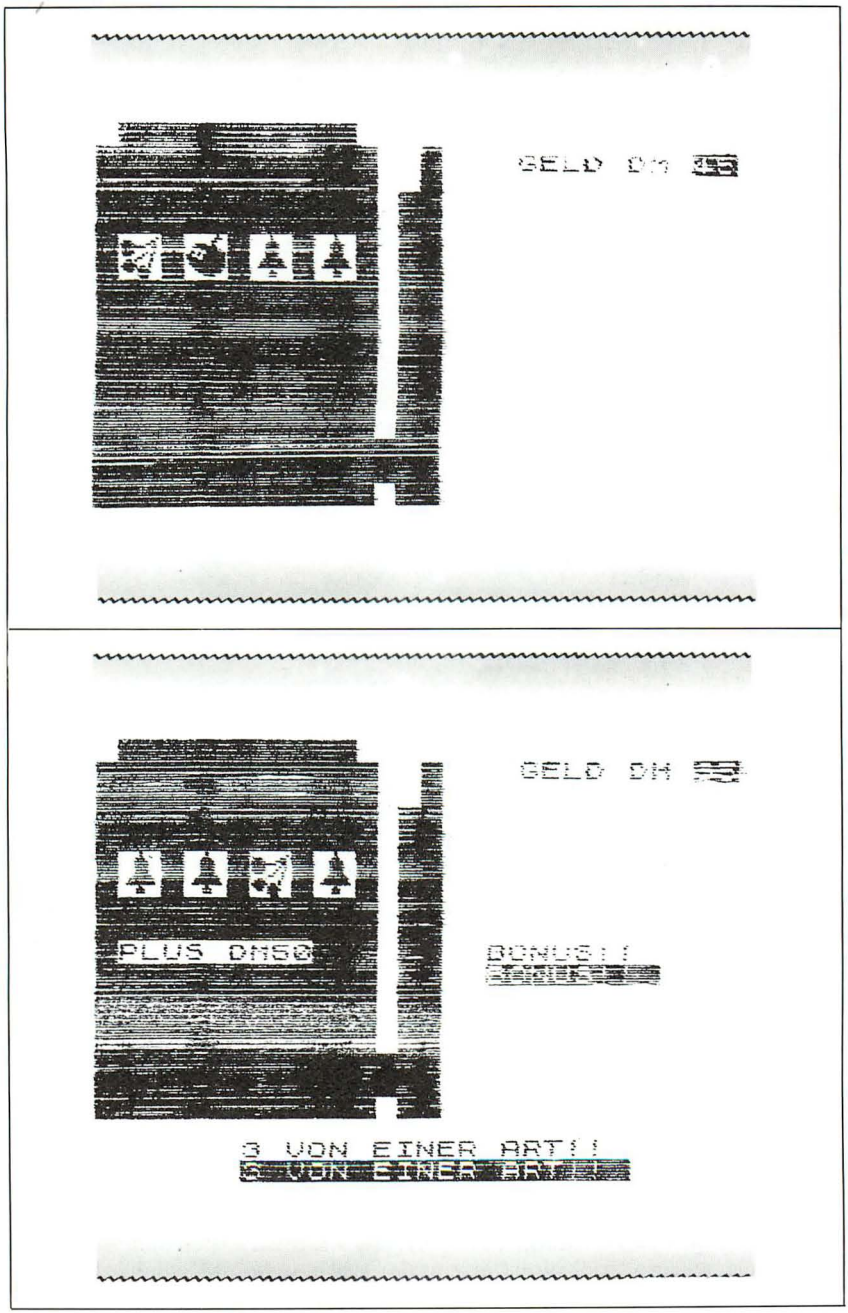
```
  10 REM SPIELFIEBER
  20 REM ***********
  30 GO SUB 720: REM VARIABLE
  40 GO SUB 290: REM WALZEN
  50 IF RND>.45 THEN GO SUB 70:
REM STOP
  60 GO TO 40
  70 REM ********************
  80 REM     STOP
  90 REM ********************
 100 FOR t=1 TO 7: BEEP .1,50-t*
7: NEXT t
 110 PRINT AT 9,22; FLASH 1; INK
 RND*4;"STOP!"
 120 INPUT q$
 130 IF q$="" THEN GO TO 190
```

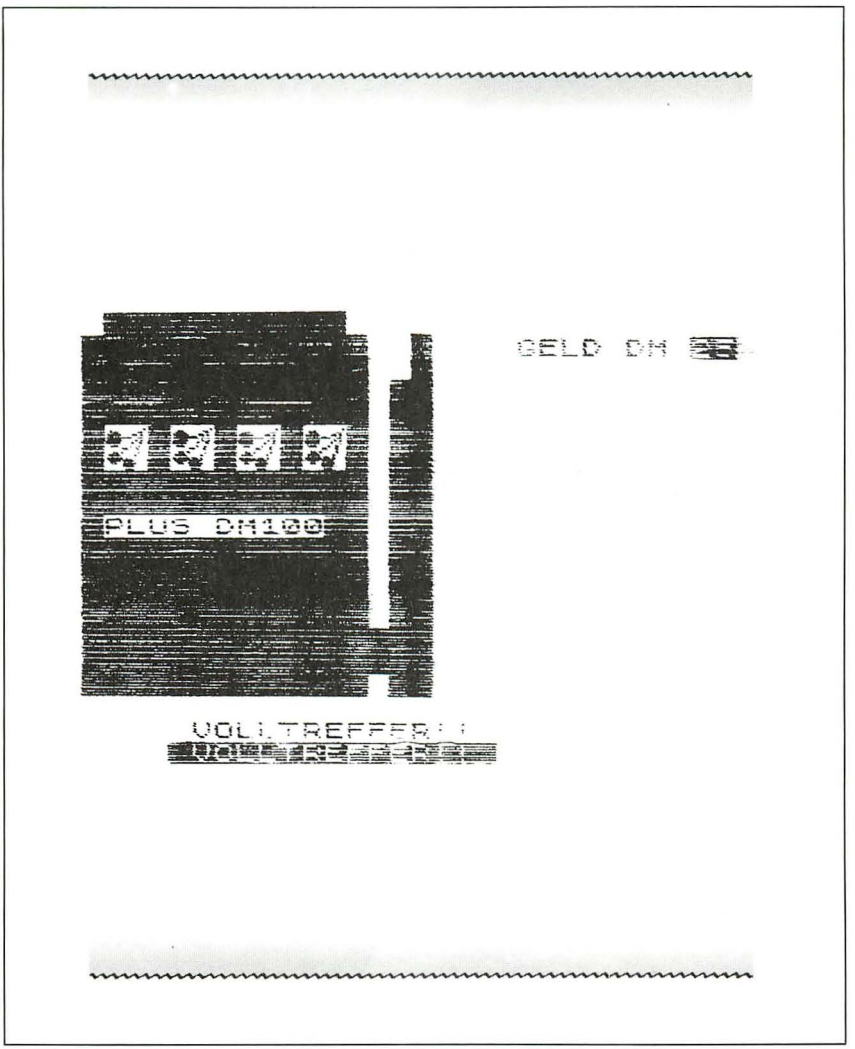
```
 140 LET q=VAL q$(1)
 150 LET a(q)=q
 170 PRINT AT 9+q,22; INK RND*4;
FLASH 1;"GESTOPPT> ";q
 180 GO TO 120
 190 PRINT AT 9,22;"          ";AT
 10,22;"          ";AT 11,22;"
          ";AT 12,22;"          ";
AT 13,22;"          "
 200 RETURN
 210 REM ********************
 220 REM PRINT WALZEN
 230 REM ********************
 240 PRINT AT 6,3;a$(a(1));AT 7,
3;b$(a(1))
 250 PRINT AT 6,6;a$(a(2));AT 7,
6;b$(a(2))
 260 PRINT AT 6,9;a$(a(3));AT 7,
9;b$(a(3))
 270 PRINT AT 6,12;a$(a(4));AT 7
,12;b$(a(4))
 280 RETURN
 290 REM ********************
 300 REM  WALZEN DREHEN
 310 REM ********************
 320 LET geld=geld-5
 330 LET preis=0
 340 LET q$="": LET p$=""
 350 PRINT AT 4,19; FLASH 1; PAP
ER 1; INK 7;"DRUECKE ENTER";AT 5
,19;"ZUM ZIEHEN";AT 6,19;"DES GR
IFFES"
 360 IF INKEY$<>"" THEN GO TO 36
0
 370 IF INKEY$="" THEN GO TO 370
 380 PRINT AT 4,19;"
 ";AT 5,19;"          ",AT 6,
19;"          "
 390 PRINT AT 2,29; FLASH 1; INV
ERSE 1; INK RND*4; geld; FLASH 0
; INVERSE 0;"  "
 400 FOR z=1 TO 8
 410 FOR b=1 TO 4
 420 IF q(b)=b THEN GO TO 450
 430 BEEP .008,z*b
 440 LET a(b)=INT (RND*3)+1
 450 NEXT b
 460 GO SUB 210
 470 NEXT z
 480 DIM q(4)
 490 IF a(1)=a(2) AND a(2)=a(3)
AND a(3)=a(4) THEN LET q$=" VOLL
TREFFER!! ": LET preis=100: GO T
O 520
 500 IF (a(1)=a(3) AND a(3)=a(4)
) OR (a(1)=a(2) AND a(2)=a(3)) O
R (a(2)=a(3) AND a(3)=a(4)) OR (
a(2)=a(3) AND a(3)=a(4)) OR (a(1
)=a(2) AND a(2)=a(4)) THEN LET q
$=" 3 VON EINER ART!! ": LET pre
is=35
 510 IF a(1)=2 AND a(2)=2 OR a(2
)=2 AND a(3)=2 OR a(3)=2 AND a(4
)=2 THEN LET p$=" BONUS!! ": LET
 preis=preis+15
 520 FOR g=1 TO 30
 530 BEEP .008,50-g: BEEP .008,5
0
 540 NEXT g
```

```
 550 LET  geld=geld+preis
 560 IF q$="" THEN GO TO 630
 570 PRINT AT 19,6; BRIGHT 1; FL
ASH 1; INK 2; PAPER 4;q$
 580 PRINT TAB 6; BRIGHT 1; FLAS
H 1; INVERSE 1; INK 2; PAPER 4;q
$
 590 IF preis>0 THEN PRINT AT 10
,3; BRIGHT 1; INK 2;"PLUS DM ";p
reis
 600 IF p$="" THEN GO TO 630
 610 PRINT AT 10,20; FLASH 1; IN
K 5; PAPER 3;p$
 620 PRINT AT 11,20; INVERSE 1;
FLASH 1; INK 3;p$
 630 FOR t=1 TO 100: BEEP .008,t
-40: NEXT t
 640 PRINT AT 2,29; FLASH 1; INV
ERSE 1; INK RND*4; geld; INVERSE
 0
 650 PRINT AT 19,6;"
   ";TAB 6;"                  "
 660 PRINT AT 10,2; INK 0;"
                  "
 670 PRINT AT 10,20;"          ",
AT 11,20;"
 680 IF  geld<1 THEN PRINT AT 0,
0; FLASH 1;"Das ist das Ende, me
in Freund","Du bist geschlagen":
STOP
 690 IF  geld >499 THEN PRINT AT
 0,0; FLASH 1;"Du hast die Bank
gesprengt!!": STOP
 700 RETURN
 720 REM ********************
 730 REM Variable/Graphik
 740 REM ********************
 750 LET  geld=50
 760 DIM a$(3,4): DIM b$(3,4): D
IM a(4): DIM q(4)
 770 FOR z=0 TO 7
 780 READ a: READ b: READ c: REA
D d
 790 POKE USR "a"+z,a
 800 POKE USR "b"+z,b
 810 POKE USR "c"+z,c
 820 POKE USR "d"+z,d
 830 NEXT z
 840 LET a$(1)=CHR$ 16+CHR$ 2+"a
"+"b"
 850 LET b$(1)=CHR$ 16+CHR$ 2+"c
"+"d"
 860 FOR z=0 TO 7
 870 READ e: READ f: READ g: REA
D h
 880 POKE USR "e"+z,e
 890 POKE USR "f"+z,f
 900 POKE USR "g"+z,g
 910 POKE USR "h"+z,h
 920 NEXT z
 930 LET a$(2)=CHR$ 16+CHR$ 6+"e
"+"f"
 940 LET b$(2)=CHR$ 16+CHR$ 6+"g
"+"h"
 950 FOR z=0 TO 7
 960 READ j: READ k: READ l: REA
D m
 970 POKE USR "j"+z,j
```

```
 980 POKE USR "k"+z,k
 990 POKE USR "l"+z,l
1000 POKE USR "m"+z,m
1010 NEXT z
1020 LET a$(3)=CHR$ 16+CHR$ 3+"j
"+"k"
1030 LET b$(3)=CHR$ 16+CHR$ 3+"l
"+"m"
1040 PAPER 7: CLS : BORDER 7
1050 PRINT AT 1,3; INK 0;"
"
1060 PRINT TAB 2; INK 0;"  "; IN
K 4; PAPER 2; FLASH 1;"
"; FLASH 0; PAPER 7; INK 0;"
 ";TAB 21; INK 2; FLASH 1;"GELD
DM ";geld
1070 PRINT TAB 2; INK 0;"  "; IN
VERSE 1; INK 4; PAPER 2; FLASH 1
;"          "; INVERSE 0; FLASH 0
; PAPER 7; INK 0;"   "
1080 FOR z=1 TO 11
1090 PRINT TAB 2;"
 "
1100 NEXT z
1110 PRINT TAB 2;" "; PAPER 6; I
NK 3; FLASH 1;"          "; FLA
SH 0; PAPER 7; INK 0;"      "
1120 PRINT TAB 2;" "; PAPER 6; I
NK 3; INVERSE 1; FLASH 1;"
     "; FLASH 0; INVERSE 0; PAPE
R 7; INK 0;"     "
1130 PRINT TAB 2;"
 "
1140 RETURN
1150 REM Apfel
1160 DATA 0,8,121,254,0,28,63,25
4,0,48,63,254,15,32,31,254
1170 DATA 31,240,15,252,51,252,7
,248,52,254,1,224,101,254,0,0
1180 REM Glocke
1190 DATA 0,128,7,240,0,128,15,2
48,1,192,15,248,3,224,31,252
1200 DATA 3,224,0,128,3,224,1,19
2,3,224,1,192,3,224,0,0
1210 REM Kirsche
1220 DATA 0,0,1,136,0,2,26,8,56,
12,60,16,124,52,126,208
1230 DATA 127,204,61,224,124,20,
25,240,124,36,1,224,56,72,0,224
1240 STOP
```

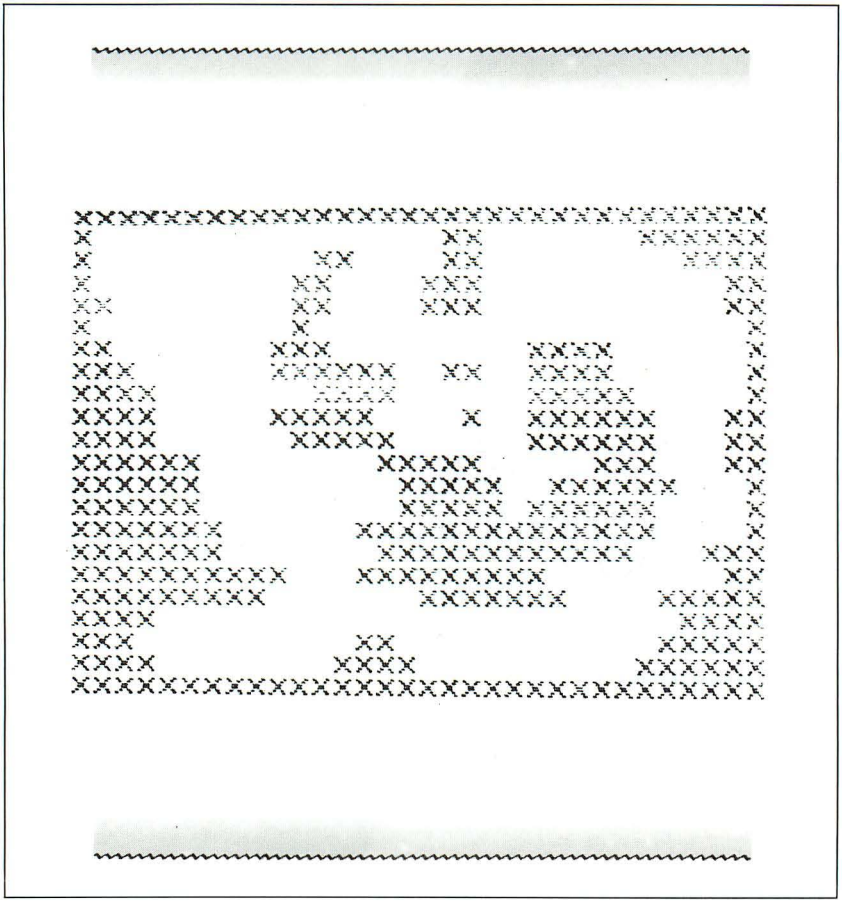
Das nachfolgende Bild zeigt ein Spielbeispiel. Sie können sich sicherlich vorstellen, wie eindrucksvoll dies mit den zusätzlichen Möglichkeiten von Farbe und Blinken aussieht!


GELD DM
GELD DM
PLUS DM50
BONUS!!
3 VON EINER ART!!



Hindernisrennen

Die SCREEN$-Funktion bildet die Grundlage für das HINDERNISRENNEN. Wir erinnern uns, daß wir mit dieser Funktion für jede Bildschirmstelle den Inhalt abprüfen können. Der Bildschirm für das HINDERNISRENNEN könnte etwa folgendermaßen aussehen:



Wir beginnen in der linken oberen Ecke des Bildschirms. Dort ist eine blinkende 9 zu sehen. Wir müssen nun im Uhrzeigersinn diese blinkende Ziffer rund um den Bildschirm führen. Hierzu benutzen wir die Tasten

```
  I
J   K
  M
```

für die vier Richtungen, da diese Tasten wie die vier Richtungen angeordnet sind. Diese Auswahl der Tasten ist etwas geschickter als die Verwendung der Tasten 5 bis 8, die normalerweise für die Führung des Cursors verwendet werden.

Wir müssen also jetzt versuchen, um den Kurs herumzufahren, ohne irgendwo anzustoßen. Wenn wir wieder zum Ausgangspunkt, zur Bildschirmecke oben links, zurückkehren, wird die Neun in eine Acht umgetauscht. Bei jedem Umlauf wird die Zahl um eins verringert. Ziel ist es also, neunmal heil herumzukommen. Wenn wir einen Unfall verursachen, gibt es Strafpunkte. Die Höhe der Punkte hängt davon ab, wie lange wir schon gefahren sind und davon, wieviel Punkte wir bisher hatten. Solange wir fahren, wird die Punktezahl ständig verringert. Wir sollten also auf jeden Fall in Bewegung bleiben und nicht zu übervorsichtig agieren. Vor dem Start sollten wir uns vergewissern, daß CAPS LOCK eingeschaltet ist, damit die INKEY$-Abfrage nicht zu umständlich wird.

```
  10 REM HINDERNISRENNEN
  20 REM ■■ CAPS LOCK EIN ■■
  30 GO SUB 360
  40 FOR z=9 TO 1 STEP -1
  50 PRINT AT autoy,autox; BRIGH
T 1; FLASH 1;z
  60 LET punkte=punkte-1
  70 LET a1=autox: LET d1=autoy
  80 IF INKEY$="J" AND autox>2 T
HEN LET autox=autox-1
  90 IF INKEY$="K" AND autox<30
THEN LET autox=autox+1
 100 IF INKEY$="I" AND autoy>1 T
HEN LET autoy=autoy-1
 110 IF INKEY$="M" AND autoy<20
THEN LET autoy=autoy+1
 120 PRINT AT d1,a1;" "
 130 IF SCREEN$ (autoy,autox)="X
"THEN GO TO 170
 140 IF autox<4 AND autoy<4 THEN
 LET autox=4: LET autoy=1: PRINT
 AT d1,a1;" ": BEEP .4,RND*50: N
EXT z
 150 IF z=0 THEN GO TO 270
 160 GO TO 50
 170 REM ***CRASH*****
 180 FOR r=1 TO 50: BORDER RND*7
 190 PRINT AT autoy,autox-1; BRI
GHT 1;"X*X";AT autoy,autox-1; IN
VERSE 1;"X*X"; INVERSE 0
 200 NEXT r: BORDER 2
 210 PRINT AT 3,5; FLASH 1;" Die
Leistung ist "; INVERSE 1;INT (p
unkte*3-127*z)
 220 FOR r=1 TO 50: BEEP .008,r/
50: BEEP .007,50/r: NEXT r
 240 CLS
 250 GO SUB 400
 260 GO TO 40
 270 REM ***ERFOLG***
 280 PRINT AT 5,0; FLASH 1; BRIG
HT 1;"Sie haben es geschafft!!!"
 290 PRINT INVERSE 1; FLASH 1; B
RIGHT 1;''TAB 5;"Bitte warten"
 300 FOR g=1 TO 300: NEXT g
 310 CLS
```

```
 320 LET punkte=10000
 330 GO SUB 400
 340 GO TO 40
 350 STOP
 360 REM **STARTWERTE**
 370 PAPER 0: INK 7: CLS
 380 BORDER 2
 390 LET punkte=5000
 400 PRINT "Anfangs-Punktestand"
; FLASH 1; Punkte: PAUSE 300
 420 FOR y=1 TO 0 STEP -1
 430 INVERSE y
 440 PRINT AT 0,0;"XXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXX"
 450 PRINT "X                  XX
     XXXXXX"
 460 PRINT "X            XX    XX
      XXXX"
 470 PRINT "X          XX     XXX
         XX"
 480 PRINT "XX         XX     XXX
        XX"
 490 PRINT "X          X
          X"
 500 PRINT "XX        XXX
 XXXX     X"
 510 PRINT "XXX       XXXXXX  XX
 XXXX     X"
 520 PRINT "XXXX        XXXX
  XXXXX   X"
 530 PRINT "XXXX      XXXXX    X
 XXXXXX  XX"
 540 PRINT "XXXX       XXXXX
 XXXXXX  XX"
 550 PRINT "XXXXXX         XXXXX
    XXX  XX"
 560 PRINT "XXXXXX          XXXXX
  XXXXXX  X"
 570 PRINT "XXXXXX          XXXXX
 XXXXXX  X"
 580 PRINT "XXXXXXX       XXXXXXX
XXXXXXX  X"
 590 PRINT "XXXXXXX        XXXXXX
XXXXX    XXX"
 600 PRINT "XXXXXXXXXX    XXXXXXX
XX       XX"
 610 PRINT "XXXXXXXXX        XXXX
XXX     XXXXX"
 620 PRINT "XXXX
       XXXX"
 630 PRINT "XXX            XX
       XXXXX"
 640 PRINT "XXXX          XXXX
        XXXXX"
 650 PRINT "XXXXXXXXXXXXXXXXXXX
XXXXXXXXXXX"
 660 NEXT y
 670 LET autox=4
 680 LET autoy=1
 690 RETURN
 700 REM I GEHT HOCH
 710 REM J GEHT LINKS
 720 REM K GEHT RECHTS
 730 REM M GEHT RUNTER
 740 STOP
```

Bei Programmbeginn wird als erstes die Routine ab Zeile 360 gestartet. Diese Routine legt die Startwerte für die Variablen fest. Sie befindet sich, wie wir das schon in früheren Programmen praktiziert haben, am Ende des Programms. Die PAPER-Farbe wird auf Schwarz gesetzt, INK auf Weiß und BORDER auf Rot. Die Ausgangspunktzahl beträgt 5000. Zeile 400 zeigt diesen Punktestand für ein paaar Sekunden auf dem Bildschirm an.

Die Y-Schleife (Zeilen 420 bis 660) zeichnet die Rennbahn zweimal. Dies gibt einen sehr eindrucksvollen Effekt bei Programmbeginn. Die Position des Autos wird durch die Variablen AUTOX für die waagerechte Richtung und AUTOY für die senkrechte Richtung festgehalten.

Nach Beendigung des Unterprogramms wird vom Hauptprogramm aus eine Z-Schleife gestartet, die von 9 bis 0 abwärts zählt. Das Auto wird in Zeile 50 ausgegeben und der Punktestand um eins verringert in Zeile 60. Zeile 70 übernimmt die Position des Autos in die Variablen A1 und D1, so daß das Auto bei Bewegung an der alten Position gelöscht wird. Die Zeilen 80 bis 110 lesen die Tastatur aus. Zeile 120 löscht die alte Position des Autos. Zeile 140 prüft, ob das Auto in der linken oberen Ecke angekommen ist. In diesem Fall geht die Steuerung zur nächsten Runde der Z-Schleife über. Zeile 150 prüft, ob die Z-Schleife abgearbeitet ist. Dann haben wir gewonnen und das Programm kann uns zu unserem Erfolg gratulieren (Zeile 270). Wenn Z noch nicht Null ist (d. h., daß der letzte Lauf noch nicht beendet ist), geht es weiter mit Zeile 160 und noch im selben Lauf der Z-Schleife zu Zeile 50. Dort wird das Auto in seiner neuen Position ausgegeben, und das Rennen beginnt von neuem.

Die Zeile 130 haben wir bei dieser Erläuterung ausgelassen. Wir müssen hierüber etwas länger sprechen, da in dieser Zeile die SCREEN$-Funktion aufgerufen und gegebenenfalls die Aufprallroutine (Zeile 170) abgerufen wird. Sehen wir uns also Zeile 130 an. Gelegentlich kann man die SCREEN$-Funktion dazu verwenden, ein Bild abzutasten und auf Kassette abzuspeichern. Jetzt wollen wir die SCREEN$-Funktion einsetzen, um zu testen, wo ein Hindernis vorliegt. In den Programmen SCHÄDELACKER und REBENKLAU haben wir davon Gebrauch gemacht.

Die beiden Zahlen, die in Klammern auf das Wort SCREEN$ folgen, haben die gleiche Bedeutung wie die Zahlen, die beim PRINT AT-Befehl zur Positionierung verwendet werden. Zeile 130 prüft die Position (AUTOY, AUTOX), zu der das Auto als nächstes fahren soll. Wenn dort ein Hindernis steht (X), kommt es zum Aufprall, noch unterstrichen durch knackende Geräusche des Fernsehers. Wenn an der fraglichen Position kein Hindernis steht, kann das Programm mit der Schleife fortfahren. Wenn Sie möchten, können Sie auch andere Hindernisse als das X bauen (allerdings müssen Sie darauf achten, daß die SCREEN$-Funktion keine Grafik-Zeichen lesen kann).

Die Aufprallroutine ist recht interessant. Die dortigen Feuerwerkseffekte lassen sich auch gut in anderen Programmen verwenden. Die R-Schleife (Zeilen 180–200) zeichnet an der Aufprallstelle ein X*X, welches sich mit *X* abwechselt. Der Bildschirmrand wird in wechselnden Farben dargestellt. Der gesamte Bildeindruck wirkt durch die dauernden Farbänderungen beunruhigend. Sobald wir die R-Schleife überstanden haben, wechselt die Farbe des Bildrandes wieder nach rot (Teil 2 – Zeile 200). Der Punktestand erscheint auf dem Bildschirm. Dieser beträgt das Dreifache des bisherigen Stands abzüglich 127 x die Nummer der Runde. Zeile 220 liefert eine Tonschleife ähnlich denen, die bereits früher in diesem Buch besprochen wurden. Zeile 250 springt zu Zeile 400, um das Gesamtbild wiederherzustellen. Hierbei wird Zeile 390 ausgelassen, um den Punktestand nicht wieder auf den Anfangswert zu setzen.

Die Routine, die den Erfolg meldet, wird nur dann abgerufen, wenn alle zehn Runden ohne Fehler absolviert wurden – also nur nach einer wirklich eindrucksvollen Leistung, die erst nach längerer Übung zu erreichen ist. „Sie haben es geschafft!" wird auf dem Bildschirm ausgegeben, und die neue Punktvorgabe wird auf 10000 gesetzt. Alles andere läuft wie gehabt.

An diesem Programm können Sie etliches ändern. Vielleicht möchten Sie das Auto als Grafik-Zeichen definieren und nur einmal herumfahren. Sie können auch versuchen, Fahrgeräusche nachzuahmen.

Der vorgegebene Parcours muß nicht strikt eingehalten werden. Änderungen sind erlaubt. Sie können sogar eine Routine schreiben, die mit jeder neuen Runde eine andere Streckenführung zeigt oder zwischen zwei verschiedenen abwechselt. Sicherlich werden Sie feststellen, daß man auch mogeln kann, indem man mit dem Auto rückwärts zur Ausgangsekke fährt und es dort so lange festhält, bis die Z-Schleife beendet ist. Schreiben Sie doch eine Routine, die diesen Fall ausschließt!

Zellteilung

ZELLTEILUNG ist eine Variante des bekannten Spiels LIFE. In der vorliegenden Version tanzen grüne Zellen über den Bildschirm und bilden Muster ähnlich wie im Programm LIFE. Das Spiel LIFE wurde von John Conway an der Universität Cambridge im Oktober 1970 erfunden. Es simuliert Leben, Wachsen und Sterben der Zellen in einer abgeschlossenen Kolonie. Das ist jedoch kein Alptraum von Malthus, in dem sich die Zellen hemmungslos ausbreiten, bis entweder die Nahrung oder der Platz ausgeht. In der Welt von LIFE wie auch in unserem Programm ZELLTEILUNG laufen Geburt, Tod und Überleben nach ganz zivilisierten Regeln ab.

Die Zellen leben auf einem Gitter und befolgen die Regeln von Conway:

- Jede Zelle auf dem Gitter hat acht Nachbarn.
- Eine Zelle kann in der nächsten Generation nur dann leben, wenn sie mindestens zwei und maximal drei Nachbarn hat.
- Wenn ein leerer Gitterplatz von genau drei Zellen umgeben ist, wird auf dem leeren Platz in der nächsten Generation eine neue Zelle geboren.
- Jede Zelle mit vier oder mehr Nachbarn stirbt in der folgenden Generation.

Wir können dieses Spiel auf verschiedene Weise schreiben. Vielleicht versuchen Sie zunächst einmal, dieses Programm selbständig zu entwickeln, bevor Sie an das vorliegende Programm gehen. Vorab brauchen Sie jedoch noch eine wichtige Information: Die aufgestellten Regeln müssen gleichzeitig für das ganze Gitter angewendet werden, so daß Reaktionen für die nächste Generation keinen Einfluß auf die Zahlen in der aktuellen Generation haben. Was wir brauchen, ist ein Spielfeld mit einem 10x10-Gitter.

Die Anfangszahlen müssen gesetzt und danach die Regeln von Conway überprüft werden. Was Sie davon brauchen, ist eine Routine, die feststellt, welche Gitterpunkte zur „Nachbarschaft" gehören.

Die nun folgende Routine können Sie als Ausgangspunkt für Ihre eigene Entwicklung benutzen oder einfach mal laufen lassen, um einen Eindruck von dem zu gewinnen, was Sie programmieren wollen.

```
  10 REM ***************
  20 REM     Zellen
  30 REM ***************
  40 GO SUB 260
  50 BEEP .008,RND*40: BORDER RN
D*7
  60 PRINT AT 2,6; INK 2; FLASH
1; BRIGHT 1;" Zellteilung   ";z;"
";AT 5,0;
  70 FOR m=2 TO 9: PRINT TAB 6;
  80 FOR n=2 TO 9
  90 LET x(m,n)=y(m,n)
 100 PRINT INK 4;CHR$ (x(m,n));"
";
 110 NEXT n
 120 PRINT
 130 NEXT m
 140 FOR m=2 TO 9
 150 FOR n=2 TO 9
```

```
 160 LET q=0
 170 FOR w=1 TO 8
 180 LET q=q+(VAL a$(w)=144)
 190 NEXT w
 200 IF x(m,n)=144 AND q<>2 AND
q<>3 THEN LET y(m,n)=32
 210 IF x(m,n)=32 AND q=3 THEN L
ET y(m,n)=144
 220 NEXT n
 230 NEXT m
 240 LET z=z+1
 250 GO TO 50
 260 REM ** STARTWERTE **
 270 RESTORE
 280 DIM a$(8,10): DIM x(10,10):
DIM y(10,10)
 290 FOR q=1 TO 8
 300 READ q$
 310 LET a$(q)=q$
 320 NEXT q
 330 DATA "X(M-1,N-1)","X(M-1,N)
","X(M-1,N+1)","X(M,N-1)","X(M,N
+1)","X(M+1,N-1)","X(M+1,N)","X(
M+1,N+1)"
 340 REM ANLEGEN EINER KULTUR
 350 FOR p=1 TO 10
 360 FOR q=1 TO 10
 370 LET x(p,q)=32: LET y(p,q)=3
2
 380 NEXT q
 390 NEXT p
 400 FOR q=1 TO 15
 410 READ a
 420 READ b
 430 LET x(a,b)=144: LET y(a,b)=
144
 440 NEXT q
 450 DATA 3,4,3,5,3,6,4,3,4,5,4,
7,5,4,5,5,5,6,6,3,6,5,6,7,7,4,7,
5,7,6
 460 FOR q=0 TO 7
 470 READ p
 480 POKE USR "A"+q,p
 490 NEXT q
 500 DATA 66,153,90,60,24,126,66
,102
 510 LET z=1
 520 RETURN
```

Lassen Sie uns das Programm jetzt einmal abschnittsweise anschauen. Wir gehen zuerst zur Routine 260 und erstellen dort die Startwerte. Zeile 280 legt drei Felder fest: A$ hält die mathematischen Beziehungen fest, die zwischen einem Gitterpunkt und den Nachbarpunkten besteht; X notiert die aktuelle Zellgeneration; Y erfaßt die nachfolgende Generation, die aus X gebildet wird.

Die Schleife von 290 bis 320 liest die DATA-Angaben der Zeile 330 und packt die Formeln zur Berechnung der Nachbarschaftspunkte in Q$. Die

Zeilen 350 bis 390 löschen das Gitter und auch das Folgegitter der nächsten Generation (32 ist der Code zum Füllen mit Leerstellen). Die Zeilen 400 bis 440 erfassen die erste Zellgeneration (DATA-Zeile 450) in beiden Gittern. Den Feldelementen wird die Zahl 144 zugeordnet. Dies ist der Code für das erste frei definierbare Grafik-Zeichen.

Die anschließende Q-Schleife, von 460 bis 490, definiert die Figur für die Zelle, die dann jeweils mit CHR$ 144 ausgegeben wird. In der ersten Generation wird die Zahl der Zellen auf eins gesetzt.

Bei der Rückkehr aus der Startwerteroutine wird ein Ton erzeugt und der Bildschirmrand auf Blinken geschaltet. Die Zahl der Zellen wird am oberen Rand des Bildschirms angezeigt, und die Zeilen 70 bis 130 geben das gesamte Gitter aus. In Zeile 90 wird zunächst die Folgegeneration (das Y-Feld) in das aktuelle Gitter (das X-Feld) übertragen.

Die nächste Schleife, von 140 bis 230, ist besonders wichtig. Ohne sie kann das Programm nicht arbeiten. Q zählt die Zellen in der Nachbarschaft eines Gitterpunktes. In Zeile 160 wird Q auf Null gesetzt. Die W-Schleife von Zeile 170 bis 190 prüft die acht benachbarten Zellen auf ihren Inhalt. Beachten Sie, daß M und N, die Kontrollvariablen der Prüfschleife, lediglich von 2 bis 9 gehen. Die Randzellen des Gitterrasters brauchen nämlich nicht auf Nachbarschaft untersucht zu werden. Zeile 180 erhöht den Wert von Q jedesmal um eins, wenn der Ausdruck VAL A$ (W) = 144 wahr ist. Der Spectrum liefert die Zahl 1 für logisch wahr. Diese Zeile ist also eine abgekürzte Schreibweise für:

```
IF VAL A$(W) = 144 THEN LET Q = Q + 1
```

Nachdem die Nachbarzellen überprüft sind, reagiert das Programm entsprechend dem Zählerergebnis. Wenn die analysierte Zelle einen Frosch enthält, d. h. den Wert 144 enthält, und weder 2, noch genau 3 Nachbarn vorhanden sind, hat Q weder den Wert 2, noch den Wert 3. Also wird die entsprechende Zelle für die nächste Generation im Y-Feld gelöscht (Wert 32). Wenn die kontrollierte Zelle aber leer ist (den Wert 32 enthält) und drei Nachbarzellen vorhanden sind (Q ist gleich 3), dann wird das Element der Folgegeneration geboren und erhält im Feld Y den Wert 144.

Nachdem dieser Vorgang für das ganze Gitter wiederholt worden ist, mit Ausnahme der Zellen des Gitterrandes, wird der Generationszähler um eins erhöht. Das Programm kehrt zurück nach Zeile 50, gibt die neue Generation aus und beginnt den Prozeß von vorn.

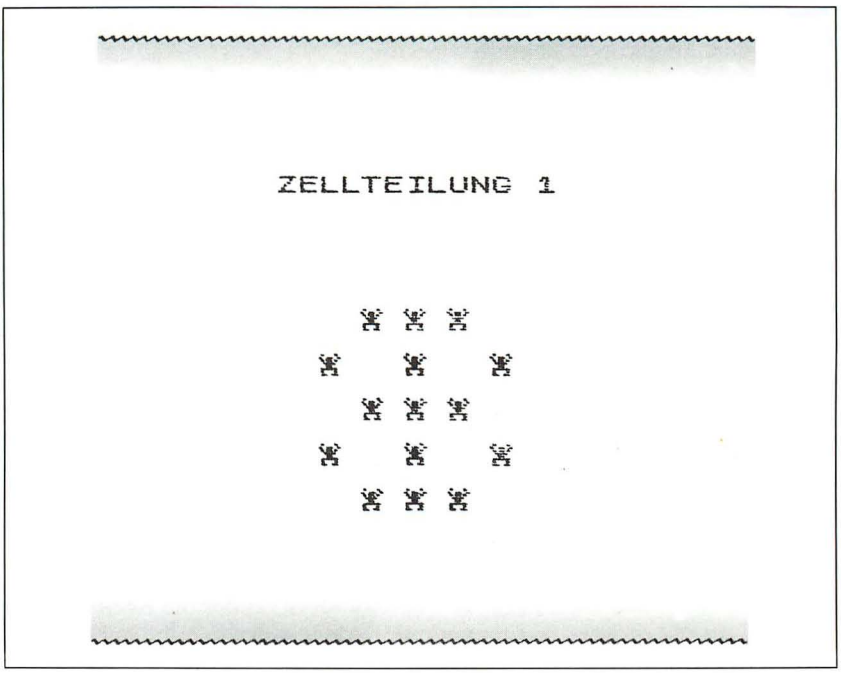
Sie können Ihre eigenen Startkolonien ins Programm einbauen. Hierzu müssen Sie lediglich die DATA-Zeile 450 ändern, in der die Koordinaten A und B für die X- bzw. Y-Elemente festgelegt werden. Vielleicht versuchen Sie einmal folgendes:

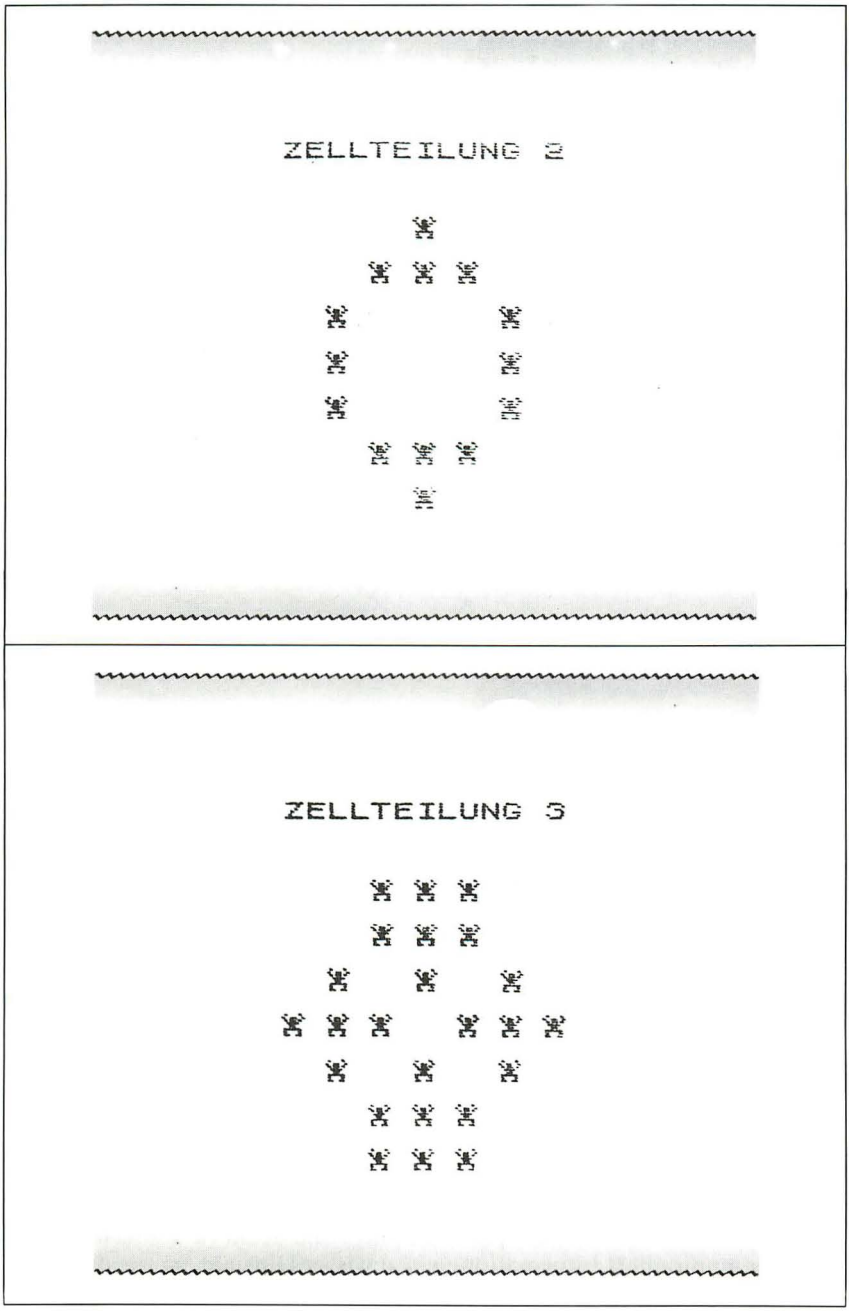
```
 450 DATA 3,3,3,5,3,6,3,8,6,3,6,
6,9,5,9,6,4,4,4,7,5,4,5,7,6,5,6,
6,7,5,7,6,8,3,8,4,8,7,8,8,9,3,9,
4,9,7,9,8
```

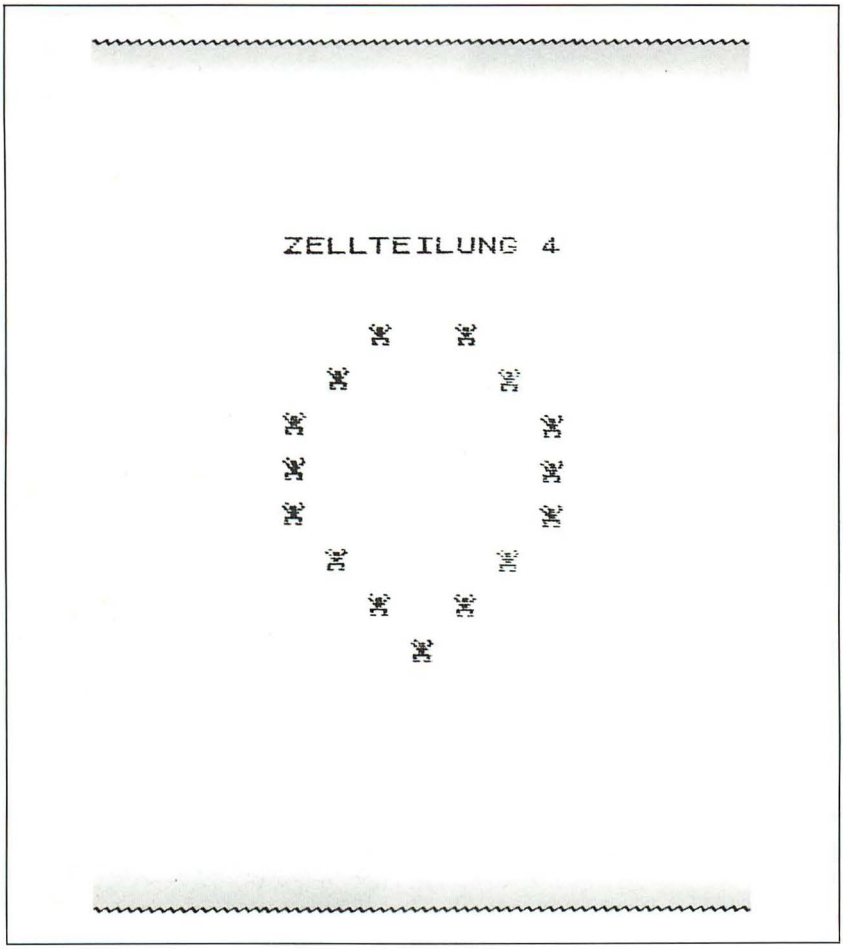
Beachten Sie bitte, daß das ursprüngliche Spiel 15 Elemente enthält. Ändern Sie daher in Zeile 400 die Zahl 15 in 24 ab. Dieses Muster lebt 16 Generationen, bevor es stirbt. Sie werden überrascht sein, was für faszinierende Muster das Programm während dieser 16 Generationen bildet. Fügen Sie noch eine weitere Zelle hinzu oder nehmen Sie eine Zelle weg. Dann können Sie beobachten, welchen Einfluß eine einzelne Zelle auf die Veränderungen des gesamten Ablaufs hat.

Vielleicht wollen Sie eine Routine schreiben, die die Startwerte aus Zufallszahlen bildet oder die Werte über Tastatureingaben hereinholt. Wenn Sie Ihre Startkolonie bilden, machen Sie sich vorab ein 10x10-Raster und numerieren Sie dieses Gitter oben und an der Seite jeweils von 1 bis 10. Dann können Sie die Koordinaten leicht ablesen. Vermeiden Sie dabei, Zellen auf den Gitterrand zu setzen.

Im folgenden sehen Sie vier mögliche Beispiele:




ZELLTEILUNG 2
ZELLTEILUNG 3



Unternehmensspiel – eine Simulation

Im UNTERNEHMENSSPIEL sind Sie der Geschäftsführer eines Betriebes, der Zibbies herstellt. Sie haben den Mitarbeiterstab fast vollständig unter Kontrolle, sowohl hinsichtlich der produzierten Menge als auch hinsichtlich des Verkaufspreises. Ihre Aufgabe ist es, den Betrieb so lange wie möglich zu führen und möglicherweise sogar eine Million Gewinn zu machen.

In diesem Programm spielt der Computer die Rolle der tatsächlichen Welt. Er manipuliert Informationen in gewisser Hinsicht, so wie dies auch in der Wirklichkeit geschehen könnte.

Zwei Schlüsselwörter in dieser Simulation sind „kopieren" und „vereinfachen". Wir versuchen, das Leben zu kopieren. Da wir jedoch nicht jede Einflußgröße erfassen können, müssen wir etwas vereinfachen, um eine überschaubare Menge von Variablen zu erhalten.

In diesem Programm, das sich während des Laufs von selbst erklärt, stehen Sie, wie wir sagten, in den Diensten einer Zibbie-Fabrik. Zu Beginn wird Ihnen mitgeteilt, wieviel Mitarbeiter Sie haben, wie hoch der Wochenlohn ist, wieviel Kapital Sie zur Verfügung haben, wie hoch Ihr Lagerbestand ist und wie hoch der Verkaufspreis für die Zibbies liegt. Das Programm nennt Ihnen jeweils die Lohnsumme. Diesen Betrag müssen Sie ständig im Gedächtnis behalten. Das Programm arbeitet in Wochen. Sie müssen dafür sorgen, daß die Lohn- und Materialkosten jede Woche bezahlt werden können. Anderenfalls müssen Sie Konkurs anmelden.

In einer Runde werden Sie zunächst einmal vor die Entscheidung gestellt, Personal einzustellen oder zu entlassen. Die Gewerkschaft wird zwar glücklich über Personaleinstellungen sein, wird jedoch bei Entlassungen Widerstand leisten. Entgegen Ihren besten Absichten können Sie also nur die Mitarbeiterzahl entlassen, die Ihnen die Gewerkschaft vorschlägt. Sie mögen zwar „7" angeben, wenn Sie nach den geplanten Entlassungen gefragt werden. Der Computer aber antwortet Ihnen: „Die Gewerkschaften gestatten Ihnen, zwei Leute zu entlassen." Damit ist gemeint, daß Sie auch im täglichen Leben keine ausschließliche Kontrolle über die Zahl der Mitarbeiter haben. Und weil jeder Mitarbeiter wöchentlich nur eine bestimmte Anzahl Zibbies herstellen kann, können Sie Ihren Betrieb nicht am Leben erhalten, wenn Sie das Personal auf eine Person reduzieren.

Von Zeit zu Zeit verlangen die Mitarbeiter eine Lohnerhöhung. Dann haben Sie keine andere Wahl, Sie müssen zahlen. Andererseits sind Ihre Zulieferfirmen bekannt dafür, beim Material maßlose Preissteigerungen durchzusetzen. Sie kämpfen also permanent gegen diese Kostensteigerungen.

Drei Faktoren können Sie in gewissen Grenzen beeinflussen:

1. Die Anzahl der Mitarbeiter (jede Entlassung kostet zusätzlich Abfindungen; Mitarbeiter zu halten, kostet ebenfalls wöchentlich einen bestimmten Betrag; die Produktion pro Kopf ist begrenzt)

2. Die Zahl der produzierten Zibbies (Sie legen das Produktionsziel fest, was aber nur selten ganz erreicht wird)

3. Der Verkaufspreis kann nach Ihrem Willen verändert werden (aber jede Änderung erschwert den Verkauf der Lagerbestände; die Zahl der Verkäufe hängt davon ab, ob Sie den Preis unverändert lassen).

Da Sie Ihre Rechnungen bezahlen müssen, müssen Sie Ihre Preise von Zeit zu Zeit den steigenden Lohn- und Materialkosten anpassen. Preisänderungen haben jedoch schwächere Verkaufsergebnisse zur Folge. Preisänderungen sollten daher nur mit Vorsicht durchgeführt werden.

Die Beschreibung mag etwas verwirrend klingen. Das macht aber nichts. Sie brauchen nicht alles auswendig zu lernen. Das Programm erklärt das meiste während des Spielverlaufs.

Das UNTERNEHMENSSPIEL hat zwei mögliche Ausgänge: Der normale ist der Konkurs. Dann wird Ihnen mitgeteilt, wieviel Wochen Sie den Betrieb am Leben erhalten konnten. Den zweiten Ausgang erreichen Sie, wenn Sie eine Million verdient haben (Kapital plus Warenbestand). In diesem Spiel sieht es sehr schlecht für Sie aus. Es ist eine alte Lebensweisheit, daß ein Betrieb zunächst das eigene Überleben sichern muß. Sie werden das im Laufe des Spiels merken. Und trotz der starken Vereinfachungen wird die Realität in diesem Spiel gut wiedergegeben. Sie werden bei einigen der Verkaufszahlen oder Produktionsergebnisse schier verzweifeln und ziemlich ratlos dastehen bei Ihren Entlassungs-/Einstellungs- und Preisfestlegungs-Entscheidungen.

Abgesehen von der Programmlänge ist das UNTERNEHMENSSPIEL tatsächlich ein relativ einfaches Spiel. Und wenn Sie erst einmal den Aufbau verstanden haben, werden Sie schnell Ihre eigenen Simulationen programmieren können.

Wie wir schon mehrfach in diesem Buch ausgeführt haben, lassen sich große Programme wesentlich leichter handhaben, wenn wir sie in einzelne Unterroutinen zerlegen und diese Unterroutinen dann nach und nach im Hauptteil des Programms aufrufen. Genauso wurde auch unser UNTERNEHMENSSPIEL entwickelt. Hier ist die Programmliste; einige Erläuterungen kommen später.

```
  10 REM **********************
  20 REM UNTERNEHMENSSPIEL
  30 REM **********************
  40 GO SUB 9000: REM VARIABLEN
  50 LET woche=woche+1
 100 GO SUB 5000: REM AUSGABE
 110 GO SUB 6000: REM ARBEITER
 120 GO SUB 5000: REM AUSGABE
 130 GO SUB 5130: REM PRODUKTION
 140 GO SUB 5000: REM AUSGABE
 150 GO SUB 4000: REM VERKAUF
 160 GO SUB 3000: REM ZUFAELLE

 170 LET kapital=kapital-lohn*an
zahl
 180 GO TO 50
2990 REM ********************
3000 REM       ZUFAELLE
3005 CLS
```

```
3010 IF RND<.45 THEN GO TO 3100
3020 LET a=INT (RND*7)+1
3025 BEEP .5,RND*50: BEEP .4,RND
*50
3030 PRINT '''"GEWERKSCHAFTSFORDE
RUNG ",a,"%LOHNSTEIGERUNG"
3040 LET lohn=INT (100*(lohn+(a*
lohn/100)))/100
3050 PAUSE 100
3060 PRINT '''"DER ARBEITSLOHN B
ETRAEGT NUN DM ";lohn
3070 PAUSE 100
3075 BEEP .5,RND*50: BEEP .4,RND
*50
3080 CLS
3100 IF RND<.85 THEN GO TO 3190
3110 PRINT ''' INK 2; FLASH 1;"E
INE STURMFLUT VERNICHTETE TEILE
IHRES LAGERS !",,,,"LAGERBERICHT
:"
3120 PAUSE 100
3130 LET a=INT (RND*lager/2)+1
3140 LET lager=lager-a
3150 PRINT '' INK 1; FLASH 1;"ZE
RSTOERTE BESTAENDE : ",a;" Zibbi
es, Wert DM ";a*verkaufspreis
3160 PAUSE 100
3170 PRINT ''"DER AKTUELE LAGERB
ESTAND IST NUN ";lager
3180 PAUSE 100
3190 IF RND>.3 THEN GO TO 3290
3195 CLS
3200 PRINT ''"DER LIEFERANT MELD
ET PREISSTEIGERUNGEN"
3210 PAUSE 100
3220 LET a=INT ((RND*100*KOSTEN/
7))/100
3225 IF a<.01 THEN GO TO 3220
3230 PRINT ''"PRODUKTIONSKOSTEN
STEIGEN UM "; FLASH 1; INK 2;"DM
";a; FLASH 0; INK 0;" PRO STUEC
K"
3240 PAUSE 100
3250 LET kosten=kosten+a
3260 PRINT '' INK 7; PAPER 1;"ZI
BBIES KOSTEN JETZT DM "; FLASH 1
;INK 7; PAPER 0;kosten;" IN DER
HERSTELLUNG": PAUSE 100
3280 IF RND<.65 AND herstellen<v
erkaufspreis THEN RETURN
3300 CLS
3305 BEEP .5,RND*50: BEEP .4,RND
*50
3310 PRINT ''"SIE KOENNEN JETZT
EINE PREIS-ERHOEHUNG VORNEHMEN"
3320 PRINT '"ZIBBIES KOSTEN JETZ
T PRO STUECK DM";verkaufspreis
3330 PAUSE 100
3340 INPUT FLASH 1;"WIEVIEL % TE
URUNGSRATE ? ";a
3345 IF a>0 THEN LET z=z+a
3350 LET verkaufspreis=INT (100*
(verkaufspreis+a*verkaufspreis/1
00))/100
3360 PAUSE 50
3370 PRINT ' FLASH 1; INK 3;"ZIB
BIES KOSTEN JETZT DM ";verkaufsp
reis
3380 PAUSE 100
```

```
3390 RETURN
3990 REM XXXXXXXXXXXXXXXXXX
4000 REM VERKAUF
4005 BEEP .5,RND*50: BEEP .4,RND
*50
4010 PRINT FLASH 1; PAPER 5;'"GE
SAMTLAGERBESTAND ";lager
4015 PAUSE 100
4020 PRINT FLASH 1; PAPER 3;'"ES
 FOLGT DER VERKAUFSBERICHT..."
4025 PAUSE 300
4030 CLS
4040 LET a=INT (RND*lager/(z/100
))+1
4045 PAPER RND*6: CLS : BORDER R
ND*6
4050 IF a>lager THEN GO TO 4040
4055 BORDER 7: PAPER 7: CLS
4060 PRINT ' INK 1;"GESAMTVERKAU
F: ";a
4070 LET lager=lager-a
4080 PRINT '"VERKAUFSVOLUMEN=DM
";a*verkaufspreis
4090 LET kapital=kapital+a*verka
ufspreis
4095 PAUSE 100
4100 RETURN
5000 REM **********************
5010 REM    AUSGABE
5020 FOR g=40 TO 50: BEEP .008,g
: BEEP .008,60-2*g: NEXT g
5022 CLS
5025 IF kapital+lager<1 THEN GO
TO 8000: REM KONKURS
5027 IF kapital+lager>999999 THE
N PRINT FLASH 1; BRIGHT 1; INK 2
;"SIE SIND MILLIONAER!!!": PAUSE
 500: GO TO 8050
5030 PRINT INK 2; FLASH 1;"GESCH
AEFTSBERICHT: WOCHE ";woche;" "
5040 PRINT ' INK 2;"LIQUIDES KAP
ITAL IST DM ";kapital
5050 PRINT ' INK 1;"IHR LAGER EN
THAELT ";lager;"ZIBBIES";TAB 10;
" MIT DEM WERT DM ";lager*verkau
fspreis
5060 PRINT "DER VERKAUFSPREIS BE
TRAEGT PRO STUECK DM ";verkaufsp
reis
5070 PRINT INK 2;"UND DIE HERSTE
LLUNGSKOSTEN BELAUFEN SICH AUF D
M ";kosten
5080 PRINT INK 7; PAPER 1,'"ANZAHL
DER MITARBEITER IST ";anzahl
5090 PRINT PAPER 1; INK 7;"DER L
OHN BETRAEGT DM"; lohn;"    ","U
ND DIE LOHNSUMME IN DIESER WOCHE
 IST DM "; lohn*anzahl
5100 PRINT INK 2;'"ANZAHL PRO PE
RSON "; produktion'" GESAMTPRODU
KTION PRO WOCHE "; produktion*an
zahl
5120 RETURN
5130 INPUT "WIEVIEL WOLLEN SIE H
ERSTELLEN?";herstellen
5135 IF herstellen=0 THEN RETURN
5140 IF herstellen*kosten>kapita
l THEN PRINT INK 2; FLASH 1;"NIC
HT GENUG GELD!": GO TO 5130
```

```
5150 IF herstellen>produktion+an
zahl THEN PRINT INK 4; FLASH 1;"
ZU WENIG ARBEITER!"  GO TO 5130
5160 PRINT AT 0,0; FLASH 1; INK
1;" ZIEL FUER WOCHE";woche;" IST
";herstellen;"          "
5170 LET herstellen=herstellen-I
NT (RND*herstellen/5*(z/100))
5180 PAUSE 100
5190 PRINT INK 3; FLASH 1;AT 0,0
;"GESAMTPRODUKTION DER WOCHE ";w
oche;" WAR ";herstellen;"       "
5200 LET lager=lager+produktion
5210 LET kapital=kapital-kosten*
produktion
5220 PAUSE 50
5300 RETURN
5310 REM ********************
6000 REM ***Mitarbeiter***
6010 INPUT "WIEVIELE MITARBEITER
 SOLLEN EINGESTELLT WERDEN ?";a

6020 LET anzahl=anzahl+a

6030 PRINT AT 0,0; FLASH 1; INK
1;"DIE ZAHL DER MITARBEITER IST
";anzahl
6035 PAUSE 100: GO SUB 5000
6037 IF a>0 THEN RETURN
6040 INPUT "WIEVIEL MITARBEITER
SOLLEN ENTLASSEN WERDEN";FLASH 1
; INK 4;a
6042 IF a=0 THEN GO TO 6090
6045 IF a>anzahl THEN GO TO 6040

6050 LET a=INT (RND*a+1)
6060 PAUSE 100
6070 PRINT FLASH 1; BRIGHT 1; IN
K 6; PAPER 2;"'''ES DUERFEN ENTL
ASSEN WERDEN:";a
6080 LET anzahl=anzahl-a

6090 PAUSE 100
6100 RETURN
7990 REM ********************
8000 REM Konkurs
8010 PRINT '''TAB 8; FLASH 1; IN
K 2;"KONKURS!!!!"
8020 PRINT '''"SO EIN PECH!"

8030 PRINT '''"ABER SIE HABEN DA
S GESCHAEFT"
8040 PRINT "INSGESAMT ", FLASH 1
; INK 1;woche; FLASH 0; INK 0;"
WOCHEN"
8050 PRINT '''" J" EINGEBEN FUER
EINEN NEUEN VERSUCH ODER "N" FUE
R AUFHOEREN!"
8055 LET a$=INKEY$
8060 IF a$="" THEN GO TO 8055
8070 IF a$="j" OR a$="J" THEN RU
N
8080 STOP
9000 REM **********************
9010 REM VARIABLE
9020 REM **********************
9030 LET kapital=500+INT (RND*50
0)
9040 LET lager=100+INT (RND*50)
```

```
9050 LET verkaufspreis=10+INT (R
ND*5)
9060 LET kosten=2+INT (RND*5)
9070 IF kosten>verkaufspreis THE
N GO TO 9050
9080 LET anzahl=7 + INT (RND*10)
9090 LET lohn=12+INT (RND*verkau
fspreis*5)
9100 LET produktion=5+INT (RND*6
)
9110 LET woche=0
9120 REM Z IST DER VERKAUFSHINDE
RNISFAKTOR
9130 LET z=1
9140 PAPER 7: CLS : BORDER 7: IN
K 0
9500 RETURN
```

Wir beginnen mit der Schleife von 40 bis 180:

- GOSUB 9000 – Hier werden die Variablen gesetzt, welche sich von Spiel zu Spiel ändern. Sie legen die Zahl der Mitarbeiter fest, den Anfangslohn, den Warenbestand, den Verkaufspreis der Zibbies usw.
- Zeile 50 erhöht die Variable WOCHE um eins.
- Das Unterprogramm ab Zeile 5000 wird mehrfach aufgerufen (in Zeile 100, 120 und 140), um die Veränderungen während der Woche zu berichten.
- Die anderen Unterroutinen werden durch die jeweiligen REM-Zeilen erklärt:
  - MITARBEITER (entlassen oder einstellen ab 6000)
  - PRODUKTION (ab 5130)
  - VERKAUF (ab 4000)
- Die letzte Routine (UNVORHERGESEHENES, ab 3000) wird in jeder Arbeitswoche einmal durchgegangen. Aber die einzelnen Abschnitte werden teilweise oder ganz übersprungen. Hierzu zählen Gewerkschaftsforderungen nach mehr Geld, oder Hochwasser, das einen Teil Ihres Warenbestandes zerstört. Hierdurch gewinnt das Spiel erheblich an Spannung. Die unvorhergesehenen Ereignisse werden jedoch nie so stark, daß sie das Können eines geschickten Spielers total übertreffen.

Wenn Sie ein Simulationsprogramm schreiben, müssen Sie erst genau überlegen, was Sie im einzelnen wiedergeben wollen, und welche Variablen Sie in das Programm einbeziehen wollen. Dann müssen Sie die Schleife des Hauptprogramms schreiben, mit der der Programmablauf kontrolliert wird. Als nächsten Schritt könnten Sie sich dann die generelle

Ausgabe überlegen (hier ab Zeile 5000), noch bevor Sie den Variablen irgendwelche Werte zuweisen. Sie werden sehen, daß sich die benötigten Variablennamen automatisch entsprechend der Ausgabe ergeben. Die Variablennamen werden ganz bewußt in Langschrift geschrieben, um das Programm lesbar und wartungsfreundlich zu gestalten. Variablennamen wie z. B. ZA3 für den Lohn oder BB für den Warenbestand werden tunlichst vermieden. Sie brauchen also kein eigenes Variablennamenregister. Derart willkürliche Variablennamen führen meist auch zu erheblichem Durcheinander bei der Programmierung. Variablen lang auszuschreiben braucht zwar länger, macht sich aber sofort bezahlt.

Da die Ausgabe den Kern der Programmentwicklung darstellt, wollen wir uns diesen Teil zuerst anschauen:

```
   VERKAUFSLISTE: WOCHE 5

ANFANGSKAPITAL IST DM2657.92

DER VERKAUFSWERT VON 12
ZIBBIES BETRAEGT DM 169.68

DER VERKAUFSPREIS IST DM 14.14
PER STUECK UND DIE PRODUKTIONS
KOSTEN SIND PER STUECK DM 7.41

ANZAHL DER MITARBEITER IST 7
IHR LOHN BETRAEGT JE DM 41 UND
DIE WOCHENLOHNSUMME IST DM287

JEDER MITARBEITER KANN 10 ZIB
BIES PRO WOCHE HERSTELLEN, AL
SO EINE GESAMTSUMME VON 70
```

Zunächst wird die Woche mitgeteilt, dann das verfügbare Kapital. Davon müssen die Löhne und das Rohmaterial bezahlt werden. Weiterhin erfahren Sie den Bestand an Zibbies, Stückzahl und Verkaufspreis sowie die Herstellkosten und den Abgabepreis pro Stück. Außerdem erfahren Sie die Zahl der Mitarbeiter, die unterschiedlichen Löhne sowie die Lohnsumme. Als letztes schließlich wird gesagt, wieviel Zibbies pro Mitarbeiter hergestellt werden können. Hierbei handelt es sich um eine Sollvorgabe, die jedoch nur selten erreicht wird.

Diese Ausgaben bzw. Varianten davon erscheinen immer wieder während des Spiels. Hierbei werden Farbe, Blinken und Aufhellen benutzt, um die Ausgabe übersichtlicher zu gestalten. Beim ersten Durchlauf werden Sie die Parameter sehen, die das Schicksal für Sie bereit hält. Danach geht das Programm in die Routine MITARBEITER, beginnend mit der Zeile 6000. „Wieviel Mitarbeiter möchten Sie einstellen?“ werden Sie gefragt. Wenn Sie beschließen, Mitarbeiter einzustellen, werden diese zum Personalbestand hinzugerechnet und in einer neuen Übersicht ausgege-

ben. Die Lohnsumme liegt jetzt höher. Die nächste Entscheidung betrifft nun die Zibbies. Wieviel davon wollen Sie herstellen?

Falls Sie jedoch keine Einstellungen planen, werden Sie gefragt „Wieviel Mitarbeiter wollen Sie entlassen?". An diesem Punkt treten die Gewerkschaften in Erscheinung und diktieren Ihnen, von wie vielen Mitarbeitern Sie sich trennen dürfen. Wieder erscheint die Bildschirmausgabe und zeigt die neue Mitarbeiterzahl und die Lohnsumme.

Wenn Sie das Mitarbeiterproblem überlebt haben, werden Sie jetzt vor Produktionsprobleme gestellt (Unterroutine PRODUKTION ab Zeile 5130). „Wieviel Zibbies möchten Sie herstellen?", werden Sie gefragt. Wenn Sie Null eingeben, erscheint wieder die Ausgabe, und das Programm wechselt über zur Verkaufsroutine, um ab Lager zu verkaufen. Wenn Sie sich jedoch entschließen, Zibbies herzustellen, vergleicht das Programm Ihr Produktionsziel

a) mit der Mitarbeiterzahl und deren Leistungsvermögen in dieser Woche,

b) mit den Materialstückkosten.

Falls Sie nicht genug Mitarbeiter haben, erhalten Sie die entsprechende Meldung und werden nach einem anderen Produktionsziel gefragt. Wenn Ihr Geld für den Materialeinkauf nicht reicht, erscheint die Meldung „nicht genug Geld". Sobald Sie eine akzeptable Zielvorgabe gemacht haben, erscheint oben auf dem Bildschirm z. B. die Meldung „Wochenziel ist 92". Nach einer kurzen Pause wird dann gemeldet, wieviel Stück tatsächlich produziert werden können: „In Woche 4 wurden 86 hergestellt."

Jetzt ist Schluß mit weiteren Erläuterungen, sonst vergeht Ihnen noch der Spaß an diesem Spiel. Lassen Sie sich beim Spiel noch ein bißchen überraschen. Wenn Sie den Erklärungen zu den anderen Programmen aufmerksam gefolgt sind, werden Sie sicherlich keine Schwierigkeiten mehr haben, dieses Spiel zu verstehen.

Dieses Programm wurde in das Buch aufgenommen, weil Simulationen ein Gebiet sind, auf dem der Computer von großem Nutzen sein kann. Simulationen, um eine Raumstation oder eine Würstchenbude zu führen, können einen Eindruck vermitteln, wie solche Probleme im täglichen Leben bewältigt werden können. In einem echten Produktionsbetrieb treten vielfältige, unvorhersehbare Ereignisse ein. So kann sich ein Arbeiter verletzen oder auf Grund der großen Hitze langsamer arbeiten, am Geburtstag aussetzen oder zum Zahnarzt gehen. Es könnten Stockungen im Nachschub oder Schäden bei den Maschinen auftreten. Ein Feuer könnte durch einen Kurzschluß ausbrechen, ein Gewerkschaftsstreik den Betrieb lahmlegen etc. Sie werden recht schnell lernen, welche Schwierigkeiten entstehen, wenn Sie die tatsächlichen Probleme detailliert übernehmen wollen.

Damit kommen wir zum Ende des Spielekapitels. Es ist das größte Kapitel in diesem Buch, weil Spielen und das Programmieren von Spielen die sichersten und anregendsten Wege sind, um Programmiertechniken zu erlernen. Außerdem bieten Spiele Entspannung nach einem harten Computereinsatz. Es bleibt zu hoffen, daß dieses Kapitel in beiderlei Hinsicht von Nutzen ist.

Vorschläge zur weiteren Lektüre:

Ahl, David (Hrsg.): BASIC Computer-Spiele, Bd. 1 und 2, SYBEX-Verlag GmbH, Düsseldorf, 1982

What To Do After You Hit RETURN, Hayden Book Company, Inc., USA, 1981

Daines, Derrik: 26 BASIC Programs for your Micro, Newnes Technical Books/Butterworth & Co., 1982

Watson, Wm. Scot: 67 Ready-To-Run Programs in BASIC, TAB Books Inc., USA, 1981

Mateosian, Richard: Inside BASIC Games, SYBEX Inc., USA, 1981

Spencer, Donald D.: Game Playing with BASIC, Hayden Book Company, Inc., USA, 1981

Lee, J. D., Beech, G., Lee, T. D.: Computer Programs That Work! Sigma Technical Press, Wolverhamton, 1980

Chisman, Margaret: Producing Computer Poetry, Originalartikel aus Creative Computing (S. 106/107) in The Best of Creative Computing, Band 2 von Ahl, David (Hrsg.), Creative Computing Press USA

Neue Ideen für Computer-Spiele:

Arnold, Peter (Hrsg.): The Complete Book of Indoor Games; Hamlyn, 1981

Brandreth, Gyles: The Complete Home Entertainer; Robert Hale, 1981

Kapitel 8

Dreidimensionale Graphik

Science-Fiction-Filme wie „Krieg der Sterne" zeigen, wie der Computer komplexe, dreidimensionale Bilder produzieren und sie in Echtzeit verändern kann. Das hier erläuterte Spectrum-Programm kann zwar keine derart komplexen Bilder herstellen, aber es produziert immerhin einigermaßen dreidimensionale Bilder, die Sie (durch Rotation) wie reale Objekte behandeln können.

Das Programm erlaubt Ihnen, die Figuren mit geraden Linien zu zeichnen, deren Länge Sie beliebig bestimmen können. Die Benutzungsanleitung zu diesem Programm mag Sie zunächst etwas verwirren, aber wenn Sie die Erläuterungen des Programms bei der Eingabe sorgfältig studieren, werden Sie das Programm sehr schnell in den Griff bekommen und schließlich recht eindrucksvolle „3-D"-Bilder herstellen können.

```
    1 REM **********************
    2 REM Dreidimensionale Ecken
    3 REM
    4 REM **********************
    5 DIM s$(255)
    6
   10 GO SUB 5000
   11 GO SUB 1000
   15 CLS
   20 GO SUB 7000
   25 IF p=1 THEN GO TO 200
   26
   40 LET r$=INKEY$  IF r$="" THE
N GO TO 40
   41 LET s$(a)=r$
   42 LET a=a+1
   43 IF a=255 THEN PRINT "SPEICH
ER VOLL"  STOP
```

```
  44 IF r$="S" THEN GO TO 10
  50 GO SUB 3000
  60 GO SUB 2000
  70 DRAW c-PEEK (23677),d-PEEK
(23678)
  80 GO TO 40
  90
 200 GO SUB 6000
 205 LET r$=INKEY$: IF r$="" THE
N GO TO 205
 210 GO TO 10
 220
1000 LET s=l*l+m*m
1010 LET t=s+n*n
1020 LET q=SQR (t)
1030 LET h=SQR (s)
1040 RETURN
2000 LET o=t-u*l-v*m-w*n
2010 LET c=t*(v*l-u*m)*4/(h*o)+1
28
2020 LET d=96+3*q*(w*s-n*(u*l+v*
m))/(h*o)
2286 RETURN
2287
3000 IF r$="O" THEN LET w=w+g
3010 IF r$="U" THEN LET w=w-g
3020 IF r$="R" THEN LET u=u-g
3030 IF r$="L" THEN LET u=u+g
3040 IF r$="Z" THEN LET v=v-g
3050 IF r$="V" THEN LET v=v+g
3060 RETURN
3999
5000 CLS
5010 INPUT "GROESSE?";G
5020 INPUT "BEOBACHTUNGSWERT X ?
";l: INPUT "BEOBACHTUNGSWERT Y ?
";m: INPUT "BEOBACHTUNGSWERT Z ?
";n: LET a=1
5061 PRINT "E)RSTELLUNG, W)IEDER
HOLUNG, Z)UFALLSZEICHNUNG"
5070 LET r$=INKEY$: IF r$="" THE
N GO TO 5070
5080 IF r$="W" THEN LET p=1: GO
TO 5110
5090 IF r$="E" THEN LET p=0: GO
TO 5110
5093 IF r$="Z" THEN GO TO 8000
5100 GOTO 5070
5110 RETURN
6000 GO SUB 7000
6010 LET r$=s$(a)
6020 GO SUB 3000
6030 GO SUB 2000
6040 DRAW c-PEEK (23677),d-PEEK
(23678)
6050 LET a=a+1
6060 IF s$(a)<>"S" AND a<>255 TH
EN GO TO 6010
6070 RETURN
6999
7000 LET w=0: LET u=0: LET v=0:
GO SUB 2000
7010 PLOT c,d
7020 RETURN
7999
8000 LET a=1: LET g=20: LET l=RN
```

```
D*100: LET m=RND*100: LET n=RND*
100
8003 CLS
8010 GO SUB 6000
8020 PAUSE 50
8030 GO TO 8000
```

Prüfen Sie, ob sich der Computer im CAPS LOCK-Zustand befindet. Halten Sie die CAPS SHIFT-Taste unten und drücken Sie auf die Taste 2, bevor Sie das Programm starten. Beim Durchlauf werden Sie als erstes die Frage GRÖSSE? am unteren Rand des Bildschirms sehen. Hier sollen Sie die Maße für die Linien eingeben, aus denen sich die Figuren zusammensetzen. Um Ihnen eine Vorstellung von dem Maßstab zu geben, den das Programm benutzt: Die Seiten des Würfels in Abb. 1 haben eine Länge von 20. Da dies ein guter Ausgangspunkt ist, geben Sie 20 ein und drücken Sie die Taste ENTER.

Die nächste Frage auf dem Bildschirm ist: Beobachtungspunkt X? Geben Sie bei diesem ersten Durchlauf 100 auf diese Frage ein und 10 bzw. 20 für die Y- und Z-Koordinaten des Beobachtungspunktes. Nachdem der Bildschirm wieder frei ist, können Sie folgendes lesen:

E Erstellung
W Wiederholung
Z Zufallszeichnung

Dies sind die drei Möglichkeiten, die Ihnen angeboten werden. Wir beginnen mit der Erstellung. Hierzu muß die E-Taste gedrückt werden. Wenn nichts passiert, haben Sie wahrscheinlich vergessen, die CAPS LOCK-Taste einzuschalten.

Der Bildschirm wird wieder gelöscht, und ein winziger Punkt bleibt im Zentrum zurück. Dieser Fleck ist der Ausgangspunkt für die Figur, die gezeichnet werden soll.

Das Programm akzeptiert jetzt sechs verschiedene Eingaben: O(ben), U(nten), L(inks), R(echts), V(orwärts) oder Z(urück). Sie müssen nur die Taste mit den Anfangsbuchstaben der gewünschten Richtung drükken. Bleiben Sie mit Ihrem Finger nicht zu lange auf irgendeiner Taste, sonst wird die Linie länger als beabsichtigt. Spielen Sie ein bißchen mit dem Programm herum und drücken Sie die verschiedenen Richtungstasten nach Belieben, dann kommen Sie wieder zurück zum Buch.

Versuchen Sie jetzt, dieses Beispiel zu zeichnen:

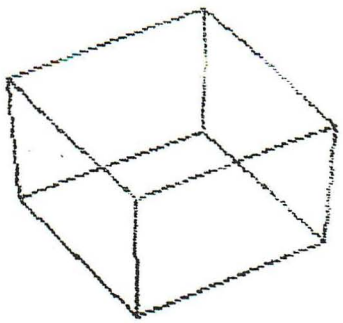


Abb. 1:

Löschen Sie zuerst den Bildschirm, entweder dadurch, daß Sie die Taste S (für Schluß) als gewünschte Richtung angeben oder dadurch, daß Sie wieder mit RUN beginnen. In beiden Fällen erscheint die Frage nach der Größe. Dann benutzen Sie die gleichen Zahlen wie bereits oben angegeben, um ERSTELLUNG/WIEDERHOLUNG/ZUFALLSZEICHNUNG auf den Bildschirm zu bekommen. Fangen Sie mit E an und los geht's.

Tippen Sie jetzt ein O. Damit wird eine Linie von Punkten nach oben gezeichnet. Wenn Sie ein L tippen, geht die Linie nach links hoch. Von dort kommt man mit U nach unten und erhält so ein Tor, welches man mit R nach rechts unten zu einem Parallelogramm schließen kann. Mit V kann man jetzt vorwärts gehen und danach wieder die gleiche Prozedur mit O, L, U, R ausführen.

Jetzt benötigen wir nur noch drei weitere Linien, um den Würfel zu vervollständigen. Wir tippen also die Tasten:

O, Z, V, L, Z, V, U, Z, V und R

Wenn Sie den Bildschirm sorgfältig beobachten, werden Sie recht schnell erkennen, wie der Spectrum Ihren Anweisungen folgt. An dieser Stelle sollten wir die Taste S tippen und die gleiche Vorgehensweise noch einmal ausprobieren, bis der Würfel fehlerfrei gelingt.

Nachdem wir die 3-D-Graphik genügend ausprobiert haben, gehen wir jetzt mit S zurück zur Frage nach der Größe und antworten wie folgt:

Größe? 10
Beobachtungspunkt X? 100
Beobachtungspunkt Y? 150
Beobachtungspunkt Z? 200

Danach wählen wir im letzten Menü die W-Möglichkeit.

Der Würfel wird nun automatisch gezeichnet, jedoch mit kleinerem Maßstab. Wenn nichts auf dem Bildschirm erscheint, haben Sie wahrscheinlich zuvor einen Programm-Neustart gemacht. Wenn Sie jedoch über S zur Abfrage nach der Größe gegangen sind, hat das Programm die alten Eingaben noch im „Gedächtnis" und kann die letzte Zeichnung jederzeit wiederholen. Bei einem Programm-Neustart ist jedoch keine „alte Zeichnung" vorhanden. Vielleicht probieren Sie noch ein bißchen mit den Tasten S, der Auswahl W und mit verschiedenen Parametern für die Größe und den Beobachtungspunkt.

Wie funktioniert das alles?

Das Programm beginnt beim Zeichnen immer mit der Koordinate 0, 0, 0, dem Ursprungspunkt des dreidimensionalen Raumes. Durch die Angabe des Beobachtungspunktes sagen Sie dem Computer, von wo aus Sie das Bild gezeichnet haben möchten. Im Menü (Erstellung/Wiederholung/Zufallszeichnung) wird danach gefragt, ob die Zeichnung über Tastatureingabe neu konstruiert werden oder ob die zuletzt eingegebene Zeichnung wiederholt werden soll. Bei der Auswahl einer Zufallsperspektive wird das bereits eingegebene Objekt von einem zufälligen Punkt aus betrachtet. Nach einer Pause wird das Objekt erneut aus einer völlig anderen Richtung betrachtet. Dieser Vorgang wird, wenn man ihn nicht unterbricht, endlos oft wiederholt. Die Auswahl Z setzt ebenso wie W ein bereits über Tastatur eingegebenes Objekt voraus. Vielleicht versuchen Sie daraufhin einmal, Abbildung 2 nachzuzeichnen.

Abbildungen 3 bis 6 zeigen eine weitere, ziemlich einfache Figur, die jeweils aus einer anderen Perspektive gezeichnet wurde. Abb. 3 zeigt das Original; bei Abb. 4 wurde der Maßstab verkleinert; Abb. 5 und 6 haben den gleichen Maßstab wie Abb. 4. Zu Abb. 5 ist die Figur von der Stirnsei-

te aus gezeichnet worden und in Abb. 6 von unten gesehen. Weil das Programm nur „Drahtgestelle" liefert, ist es etwas schwierig zu entscheiden, ob die Figur 6 von oben oder von unten zu sehen ist.

Nach einigem Probieren werden Sie herausfinden, daß komplexe Gebilde am deutlichsten bei einer Schrittgröße von 10 herauskommen.

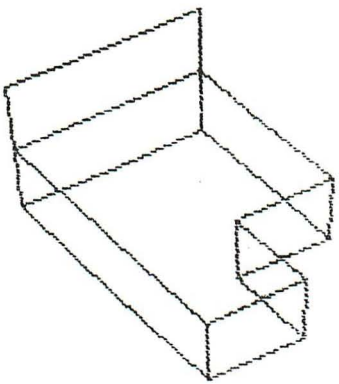


Abb. 2:

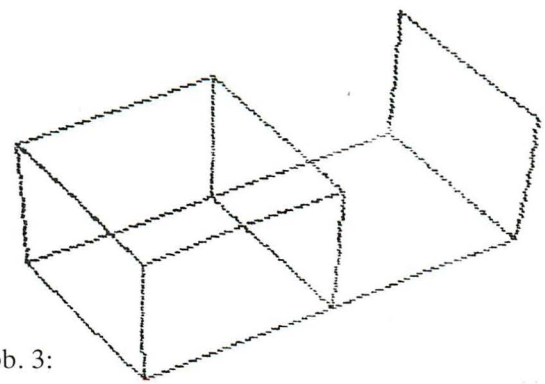


Abb. 3:

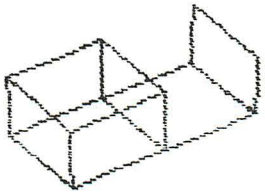


Abb. 4:

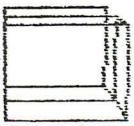


Abb. 5:

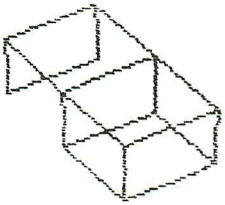


Abb. 6:

Programmerläuterungen:

Das Programm besteht aus einer Reihe von Unterprogrammen, die vorab erklärt werden. Danach folgen die Erläuterungen zu den Zeilen 1 bis 210.

Unterprogramm 1000 – neuer Ausgangspunkt.
Dieses Unterprogramm bestimmt mit L, M und N den neuen Standpunkt. Hierdurch werden auch die Variablen S, T, Q und H bestimmt. Diese Variablen werden später im Zeichen-Algorithmus verwendet.

Unterprogramm 2000 – Berechnen zweidimensionaler Koordinaten.
Dieses Unterprogramm gibt in C und D die Bildschirmkoordinaten wieder, die dem Punkt U, V, W entsprechen. L, M und N und alle Koordina-

ten, die im Unterprogramm 1000 definiert sind, werden auch im Unterprogramm 2000 benutzt. Die Konstanten 128 und 96 in den Zeilen 2010 und 2020 haben die Aufgabe, das Bild in eine zentrale Lage zu bringen, indem sie sicherstellen, daß sich der Punkt 0, 0, 0 im Mittelpunkt des Bildschirms befindet.

Unterprogramm 3000 – Änderung der Koordinaten.
Jedes Mal, wenn die Richtungstasten benutzt werden, werden die augenblicklichen Koordinaten U, V, W um einen bestimmten Wert (G) vergrößert oder verkleinert. Das Unterprogramm 3000 führt diese Operation beim Druck auf die R-Taste aus.

Unterprogramm 5000 – Menü.
Dieses Unterprogramm dient zur Festlegung der Startwerte. In den Zeilen 5010 bis 5020 werden die schon bekannten Anfangsfragen gestellt. Danach wird mit Zeile 5061 das Menü gezeigt und in Zeile 5070 notiert, was Sie ausgewählt haben. Die Variable A in Zeile 5020 zeigt die Zeichenkette, in der jede gedrückte Taste für die W- und Z-Auswahl gespeichert ist. Die Zeilen 5080 bis 5093 entschlüsseln den jeweiligen Tastendruck. Wenn W gewählt wird, wird P als wahr festgelegt, wenn E gewählt wird, wird P als falsch festgelegt. Bei der Auswahl von Z wird das Unterprogramm ab 8000 aufgerufen.

Unterprogramm 6000 – Automatisches Zeichnen.
Dieses Unterprogramm zeichnet selbständig die Figur, die in S$ gespeichert ist. Es wird bei der W-Auswahl benutzt.

Zeile 6000: ruft das Unterprogramm 7000 ab, welches den schwarzen Ausgangspunkt in den Mittelpunkt des Bildschirms setzt.
Zeile 6010: holt das nächste Zeichen von S$ nach R$, für den Abruf von UP 3000 in Zeile 6020.
Zeile 6040: zeichnet eine Linie zu diesen absoluten Koordinaten. Die zwei PEEKs greifen die augenblicklichen x- und y-Koordinaten des Spectrum ab. Wenn diese Koordinaten von den absoluten Koordinaten des Zielpunktes subtrahiert werden, dann kommen Sie zu einer absoluten, statt zu einer relativen Zeichnung. Diese Technik kann bei jedem Programm angewendet werden, bei dem die normale Form des DRAW-Befehls ungeeignet ist.
Zeile 6050: läßt den Zeiger auf S$ vorrücken, so daß das nächste Zeichen geprüft werden kann.
Zeile 6060: Falls das Zeichen ein S und damit das Ende des Zeichnungsvorgangs erreicht ist, springt das Unterprogramm mit RETURN zurück. Das gleiche passiert, falls der String S$ beim Zeichnen der nächsten Linie zu Ende ist.
Zeile 6070: bildet eine Schleife, um den Rest der Zeichen in S$ abzuarbeiten.

Unterprogramm 7000 – setzt die augenblicklichen Koordinaten auf 0, 0, 0 und zeichnet dort einen Punkt.

Unterprogramm 8000 – legt A, G (die Größe), L, M und N fest und zeichnet dann die Figur (Zeile 8000). Es macht eine PAUSE und geht dann zurück, um die Figur mit anderen Koordinaten wieder von neuem zu zeichnen.

Zeile 5: Dimension S$
Zeile 10: ruft Unterprogramm 5000 ab.
Zeile 11: ruft Unterprogramm 1000 ab.
Zeile 15: löscht den Bildschirm.
Zeile 20: ruft Unterprogramm 7000 ab.
Zeile 35: geht zur Zeile 200 für den Fall, daß die Auswahl W getroffen wurde.
Zeile 40: dient zur Eingabe einer Richtungstaste.
Zeile 41: speichert den Tastendruck in S$.
Zeile 42: erhöht den Zeiger A.
Zeile 43: stoppt das Programm, falls S$ ausgeführt wurde, d.h., falls bereits mehr als 255 Tasten getippt wurden.
Zeile 44: kehrt zum Ausgangsmenü zurück, wenn die Taste S gedrückt wurde.
Zeile 50: reagiert auf den Druck der Tasten.
Zeile 60: entschlüsselt die Koordinaten und
Zeile 70: zeichnet zur nächsten Position (siehe Beschreibung für Zeile 6040).
Zeile 80: macht eine Schleife, um weitere Tastenangaben aufzunehmen.
Zeile 200: ruft das Unterprogramm 6000 auf, um die W-Eingabe abzuwickeln. Danach wird in Zeile 205 auf eine weitere Eingabe gewartet, um schließlich in Zeile 210 wieder das Hauptmenü vorzulegen.

Kapitel 9

Erläuterungen zum Maschinen-Code

Vielleicht erinnern Sie sich, daß wir am Anfang dieses Buches bei der Klassifizierung von Computersprachen von Ebenen, sog. Levels, gesprochen haben, mit höheren Programmiersprachen, wie z. B. BASIC, und solchen, die der tatsächlichen Sprache des Mikroprozessors, dem „denkenden" Teil Ihres Computers, näher stehen. Eine solche Sprache nennen wir Maschinensprache.

Wenn Sie Ihren Spectrum in BASIC programmieren, braucht ein Teil der Anlage eine ganze Menge Rechenzeit, um Ihr BASIC in Maschinensprache zu übersetzen, damit der Microprozessor versteht, was Sie von ihm wollen. Aus diesem Grund ist BASIC auch eine relativ langsame Sprache. Wie wir nun demonstrieren werden, ist die Maschinensprache sehr viel schneller.

Geben Sie doch einmal das folgende Programm ein:

```
0   CLEAR 32000
20  RESTORE
30  FOR a=0 TO 15: READ x:POKE32000+a,x: NEXT a
40  DATA 33,255,63,01,01,24,22,255,35,11,120,177,200,114,24,248
50  CLS : PAUSE 20 : RANDOMIZE USR 32000 : PAUSE 20 : GO TO 50
```

Dies demonstriert, wie schnell in Maschinensprache der Bildschirm gefüllt werden kann. Vergleichen Sie dies mit einem entsprechenden BASIC-Programm:

```
10  FOR a = 1 TO 704 : PRINT "*"; : NEXT a
```

Anmerkung: Es wäre nicht fair, Längenvergleiche zur Beurteilung heranzuziehen. Das erste Programm ist sehr viel länger als nötig (wie wir später sehen werden).

Der Computer selbst versteht nur lange Ketten von Ziffern. Für den Z80 Microprozessor, der sich in Ihrem Spectrum befindet, bedeutet z. B. die Ziffernfolge 00111101 eine Anweisung. Für uns bedeutet sie absolut nichts. Beachten Sie jedoch, daß der Z80 mit 8-ziffrigen Codes arbeitet, d. h. die größte Zahl, die er damit verarbeiten kann, beträgt 255 dezimal. Die „Folge" von Einsen und Nullen stellt eine Dualzahl dar. Während eine Zahl in unserem Dezimalsystem Ziffern von 0 bis 9 kennt (Basis 10), kennt eine Zahl im Dualsystem (Basis 2) nur die Ziffern 0 und 1. Betrachten wir eine Zahl zur Basis 10, so stellt die am weitesten rechts stehende Ziffer die Anzahl der Einer (10 hoch 0), die nächste Ziffer die Anzahl der Zehner (10 hoch 1), die nächste Ziffer die Anzahl der Hunderter (10 hoch 2) dar usw.

302 ist also $(3*100)+(0*10)+(2*1)$.

Dualzahlen (Basis 2) arbeiten nach demselben Prinzip. Die am weitesten rechts stehende Ziffer ist 2 hoch 0, die nächste 2 hoch 1, die nächste 2 hoch 2.

101 dual ist also $(1*4)+(0*2)+(1*1) = 5$ dezimal.

Nehmen wir nun 11111111, die höchste Zahl, die der Z80 verarbeiten kann, so sieht dies folgendermaßen aus:

```
  1 mal   1 entspricht 2^0
+ 1 mal   2 entspricht 2^1
+ 1 mal   4 entspricht 2^2
+ 1 mal   8 entspricht 2^3
+ 1 mal  16 entspricht 2^4
+ 1 mal  32 entspricht 2^5
+ 1 mal  64 entspricht 2^6
+ 1 mal 128 entspricht 2^7
```

ergibt 255 dezimal

Wenn wir uns mit Dualzahlen befassen, ist es vielleicht ganz sinnvoll, noch einige Begriffe zu erklären, um spätere Irrtümer zu vermeiden. Bei Computern ist oft die Rede von sogenannten Bits und Bytes. Ein Bit ist eine der Ziffern einer Dualzahl, z. B. besteht die Zahl 00111010 aus acht Bits. Wir können uns mit Teilen einer Dualzahl befassen, indem wir uns auf die Bits beziehen, die sie enthält. Bit 0, das ganz rechte, ist in unserem Beispiel 0, während Bit drei, das vierte von rechts, 1 ist. Byte ist der Fachausdruck für eine Zahl aus acht Bits: 01110010 ist ein Byte.

Ein Bit hat den Maximalwert 1, der höchste Wert eines Byte ist 255 dezimal. Wir benutzen 8- und nicht 6- oder 10-stellige Maschinencodes, weil der Z80 Microprozessor als 8-Bit-Prozessor ausgelegt ist, d. h. er kann nicht mehr und nicht weniger als 8 Bit lange Informationen verarbeiten.

Da dem so ist, sind wir, was Daten betrifft, beschränkt auf ein Maximum von 255. Sie können dies sehr einfach testen, indem Sie POKE 100,256 in Ihren Rechner eintippen. Der Computer wird eine ERROR-Meldung ausgeben.

Wir wollen uns nun einmal anschauen, wie wir mit POKE eine Folge von Befehlen in den Computer bringen können. Wir starten mit der Adresse eines bestimmten Byte im Speicher und füllen dieses mit einem bestimmten Wert. Den Speicher stellen wir uns einfach vor als eine Kette kleiner Kästchen, von denen jedes eine eigene Dualzahl (sozusagen als Adresse) besitzt und einen Inhalt.

Wie eingangs erwähnt, kann mit nur einem Byte, also 8 Bits, als größte Zahl die (11111111) 255 dargestellt werden. Damit können Sie zwar auskommen, was den Inhalt angeht, aber stellen Sie sich einmal vor, es gäbe nur Hausnummern zwisch 0 und 255! Zum Glück haben wir Straßennamen zur weiteren Kennzeichnung, und so ähnlich behandeln wir Speicheradressen, nur daß wir anstelle des Straßennamens wiederum eine Zahl einsetzen. Haben wir z. B. die ersten 256 Adressen (0–255) abgezählt, kommen wir in die nächste „Straße" (Straße Nr. 1) mit wiederum 256 Adressen. Eine komplette Adresse sieht also z. B. wie folgt aus:

Straßenname	Hausnummer
00010110	01011010

Mit 256 Straßennamen à 256 Hausnummern haben wir nun insgesamt 256 x 256 = 65536 Adressen.

Rein technisch gesehen wird dies eine 16-Bit-Zahl genannt, und dies hat viele Vorteile, besonders da wir sie uns auch tatsächlich als eine Zahl von 16 Bit Länge vorstellen können. Beispiel: Wenn wir zu der 8-Bit-Zahl 10101101 eine weitere 8-Bit-Zahl, nämlich 10010001 addieren wollen, so gibt das einen Überlauf, denn das Ergebnis wäre 9 Bits lang:

```
                10101101
               +10010001
               ---------
Überlauf: →    1:00111110  ← 8-Bit-Zahl
```

Wenn wir ein Rechenwerk mit 16 Bit benutzen, gibt es kein Überlaufproblem:

```
  00000000  10101101
+ 00000000  10010001
-----------------------
  00000001  00111110
```

Wir können also fröhlich drauflos z. B. zwei Zahlen miteinander mutliplizieren, wenn deren Ergebnis nicht größer wird als 65635 (die Überlaufgrenze für 2-Byte-Zahlen).

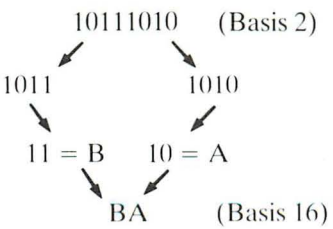
Um nun zurück auf die Speicheradressen zu kommen: Hier haben wir den Grund gefunden, weshalb der Z80 16-Bit-Adressregister benutzt, nämlich um 65635 Speicheradressen zu verwalten. Dies ergibt, durch 1024 (= 1 Kilobyte oder 1 K) dividiert, 64 Kilobyte. 64 K ist also die maximale Speichergröße, die der Z80 verwalten kann. Ihr ZX Spectrum besitzt dieses Speichermaximum durch seine 48 K RAM und 16 K ROM.

RAM steht übrigens für „Random Access Memory", was soviel bedeutet wie Direktzugriffsspeicher, ein Speicher, in den sowohl Informationen hineingeschrieben als auch herausgelesen werden können. ROM steht für „Read Only Memory", ein Speicher, der Informationen enthält, die zwar gelesen, aber nicht überschrieben werden können.

Für den gewöhnlichen Gebrauch ist der Umgang mit Dualzahlen viel zu kompliziert. Hier wird daher unsere dritte und gebräuchlichste Basis eingeführt, die Basis 16. Mit ihr kann eine 8-ziffrige Dualzahl zweiziffrig (zur Basis 16) dargestellt werden. Eine solche Zahl wird auch Hexadezimalzahl genannt. Sie erinnern sich sicher, daß eine Dualzahl die Ziffern 0 und 1 und eine Dezimalzahl die Ziffern 0 bis 9 besitzt. Demzufolge sollte also eine Zahl zur Basis 16 die Ziffern von 0 bis 15 besitzen. Nun, leider haben wir keine einstelligen Ziffern, die größer als 9 sind. Wir helfen uns daher mit den ersten sechs Buchstaben des Alphabets:

10 = A
11 = B
12 = C
13 = D
14 = E
15 = F

Die einfachste Art, eine 8-Bit-Dualzahl in eine Hexadezimalzahl oder Zahl zur Basis 16 zu konvertieren, ist die, die 8 Bits einfach in zwei Teile à vier Bits zu zerlegen und von jeder Gruppe getrennt den Zahlenwert zu bestimmen, z. B.:



Wir können nun jeden Wert eines Byte in eine 2-ziffrige Hexadezimalzahl zerlegen. Dies ist sehr bequem und stellt den einfachsten Weg dar, ein einzelnes Byte gut lesbar darzustellen.

Die folgende Tabelle zeigt die Äquivalente einer Dual-, Dezimal- und Hexadezimalzahl:

Dezimal	Dual	Hexadezimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Nachdem wir nun die mathematischen Grundlagen der Maschinensprache behandelt haben, können wir uns dem programmtechnischen Teil zuwenden.

Der Z80 versteht nur Zahlencodes anstelle von wörtlichen Anweisungen. Wir schreiben also zuerst unser Programm mit abgekürzten, aber leicht lesbaren Begriffen und setzen diese dann in Zahlen oder Zahlengruppen um, bevor wir sie in dezimaler Form mit dem Befehl POKE in den Speicher eingeben und durch eine USR-Routine ausführen.

Geben Sie doch einmal folgendes in Ihren Spectrum ein:

```
CLEAR 32000
```

Es bewirkt das Löschen des alten Speicherinhalts von 32000 bis RAM-Ende (32767 bei 16 K und 65535 bei 48 K) und schützt unsere Routine vor dem Überschreiben durch BASIC oder Löschen durch NEW. 32000 ist auch die erste Speicherstelle, an der unsere Routine sicher abgelegt werden kann.

Das nächste Problem ist nun, wie wir unseren Maschinencode in diesen Speicherbereich übertragen. Wir benutzen dafür den Befehl POKE.

Versuchen Sie dieses:

```
POKE 32000,1 (dann ENTER) drücken)
POKE 32001,0
POKE 32002,0
POKE 32003,201
```

Nun geben wir PRINT USR 32000 ein und der Rechner wird 0 ausgeben. Zahlen dieser Art zu POKEn, mag bei einigen wenigen Zahlen ideal sein, aber bei 30 oder mehr Zahlen beginnt es mühselig zu werden. Es gibt zwei Arten, dies zu vereinfachen:

1. mit einer FOR/NEXT-Schleife

```
10  FOR a=32000 TO 32003: INPUT x:POKE a,x:NEXT a
```

2. oder durch DATA-Zeilen

```
5   RESTORE
10  FOR a=32000 TO 32003: READ x:POKE a,x:NEXT a
20  DATA 1,0,0,201
```

Beide Methoden haben besondere Vorzüge hinsichtlich ihrer Anwendbarkeit. Die zweite Methode mit Hilfe der DATA-Zeilen ist jedoch vorzuziehen, weil sie leichter überprüft und geändert werden kann.

Hier ein weiteres Programm, um Maschinencodes in den Spectrum zu laden:

```
10  INPUT "Anfangsadresse?";start
20  CLEAR start
25  RESTORE
30  LET start = 1+ (PEEK 23730 + 256 * PEEK 23731)
40  FOR a = start TO 65535 (beim 16K Spectrum = 32767)
50  READ x : IF x = 999 THEN STOP
60  POKE a,x
70  NEXT a
80  DATA ...hier wird die Routine eingetragen...
```

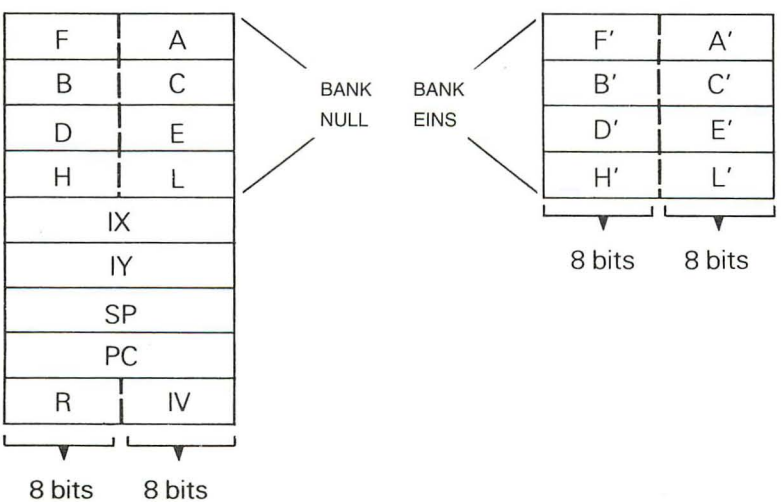
Schauen wir uns dieses Programm einmal Zeile für Zeile an. Zeile 10 fragt nach der Startadresse für das Programm. Zeile 20 (CLEAR) löscht den RAM-Speicher von der Startadresse an aufwärts. Zeile 25 (RESTORE) stellt den DATA-Zeiger auf das erste DATA-Element. Zeile 30 weist der Variablen „start" die Anfangsadresse für die Maschinenroutine zu, da sie

durch das CLEAR-Kommando gelöscht wurde. Zeile 40 kennzeichnet eine Schleife zwischen der eingegebenen Startadresse und der letzten RAM-Adresse (32767 bei 32K, 65535 bei 48K). Zeile 50 liest das erste Element der DATA-Zeile und fragt, ob dessen Wert 999 ist, den wir genommen haben, um das Ende der Daten zu kennzeichnen. Zeile 60 packt mit POKE den Wert x, der aus den DATA-Elementen gelesen wurde, an die entsprechende Adresse. Zeile 70 wiederholt die Schleife. Zeile 80 enthält die Dezimalwerte für alle Anweisungen der Routine und als letztes die Zahl 999. Es ist nicht notwendig, alle DATA in einer Zeile unterzubringen. Man kann so viele Elemente in einer Zeile unterbringen, wie man will.

Diese Routine lädt nun den Maschinencode ins RAM. Wenn wir unsere Routine ausführen wollen, fügen wir entweder die Zeile PRINT USR start oder RANDOMIZE USR start in unser Programm ein, hierbei enthält „start" die Anfangsadresse der Maschinenroutine. USR ist ein BASIC-Befehl und ist die Kurzform für UserSubRoutine, was soviel bedeutet wie Unterprogramm des Anwenders.

Im Maschinencode gibt es keine Variablen wie z. B. im BASIC. Stattdessen gibt es Register, die geladen, gelesen und verändert werden können.

Hier ist eine einfache Skizze der Z80-Register:



Jedes dieser durch einen Buchstaben dargestellten Register wie B,C,B' repräsentiert ein 8-Bit- bzw. 1-Byte-Register, welches durch Kombination mit seinem Nachbarn als Teil eines 16-Bit-Registers arbeiten kann. Alle Register in Bank 0 außer F können beliebig verwendet werden.

Die Registerpaare IX und IY sind Indexregister, die wir bei unseren Betrachtungen erst einmal außer acht lassen wollen. SP als Stackpointer wird ebenfalls vom Computer verwendet, kann jedoch nur von erfahrenen Programmierern vorteilbringend eingesetzt werden.

PC ist der Programmzähler, der die Speicheradresse festhält, an der der zuletzt ausgeführte Befehl stand. Register R wird ebenfalls vom Computer intern genutzt und hat demzufolge zeitweilig einen Zufallswert. Es ist zwar möglich, diesen Wert zu verändern, jedoch aus oben genannten Gründen nicht empfehlenswert.

Die Register in Bank 1 können zwar frei verwendet werden, sind jedoch nicht immer leicht zu handhaben. Wir werden uns daher bei unseren Anwendungen auf die Register in Bank 0 beschränken (A,B,C,D,E,H,L, ohne F).

Wie schon vorher erwähnt, handelt es sich bei diesen Registern um 8 Bit= 1-Byte-Register mit Maximalwerten bis 255. Der Z80-Maschinencode ist jedoch so konzipiert, daß es einfach ist, durch Kombination von B und C, D und E oder H und L (BC, DE, und HL) Registerpaare zu bilden, mit denen man dann wie 16-Bit-Register arbeiten kann. Wie schon vorher erwähnt, haben wir damit einen Adressierungsbereich von Null bis 65535.

Von den Registerpaaren ausgehend wenden wir uns nun der Erläuterung zu, wie wir damit arbeiten können. Um ein einzelnes Register mit einer Zahl zu laden, benutzen wir die Anweisung: LD Register, Zahl.

Wenn wir nun also das Register A (den Akkumulator, wie er genannt wird) mit dem Wert 1 laden wollen, so lautet die entsprechende Maschineninstruktion:

```
LD A,01
```

Das gleiche können wir nun auch mit den einfachen 8-Bit-Registern A, B, C, D, E, H oder L machen:

```
LD B,255
```

Die Zahl, die in das Register geladen wird, muß einen Wert zwischen Null und 255 enthalten.

Wenn wir nun D und E als Registerpaar (DE) behandeln, arbeiten wir analog:

```
LD DE,1025
```

In diesem Falle können wir jede Zahl zwischen Null und 65535 laden.

Nun haben wir eine Anweisung der Form LD B,255 zuerst in einen Dezimalcode umzuwandeln, bevor wir den Befehl in den Computer packen.

Im folgenden finden Sie eine Liste der LD-Commandos sowie ihrer Codes, wobei xxx eine Zahl zwischen Null und 255 darstellt:

LD A,xxx	62 xxx
LD B,xxx	06 xxx
LD D,xxx	22 xxx
LD E,xxx	30 xxx
LD C,xxx	14 xxx
LD H,xxx	38 xxx
LD L,xxx	46 xxx
LD BC,xxx xxx	01 xxx xxx
LD DE,xxx xxx	17 xxx xxx
LD HL,xxx xxx	33 xxx xxx

Nun sind wir nicht nur in der Lage, ein Register mit einer Zahl zu laden, sondern auch mit dem Wert eines anderen Registers. So bedeutet z. B. LD A,B, daß das Register A mit dem Wert des Registers B geladen werden soll. Unglücklicherweise gibt es kaum Befehle, um ein Registerpaar mit dem Inhalt eines weiteren zu laden. Will man also HL mit BC laden, muß man dies über folgenden Umweg tun:

```
LD H,B
LD L,C
```

Beachten Sie, daß es im Maschinencode keine Zeilennummern im eigentlichen Sinne gibt. Wenn Sie also verzweigen oder eine Schleife bilden wollen, müssen Sie eine Speicheradresse ansprechen. Bevor wir nun irgendwelche Verzweigungen durchführen, gibt es eine Menge weiterer Dinge, die man ganz einfach mit Registern durchführen kann.

Wir können z. B. eine Zahl vom Register A durch den Befehl

```
SUB A,01
```

subtrahieren. Ein entsprechender BASIC-Befehl würde lauten: LET A=A−1. Man kann natürlich ein Register von A subtrahieren:

```
SUB A,B
```

Eine komplette Liste aller Anweisungen finden Sie im Spectrum-Handbuch. Sie können sie aber auch der gängigen Literatur über Z80-Maschinensprache entnehmen.

Nach dem Aufruf einer Maschinenunterroutine, d. h., nachdem Sie in den Computer PRINT USR bzw. RANDOMIZE USR eingegeben haben, benötigt der Rechner noch eine Anweisung in Maschinencode, die ihm mitteilt, daß er nach dessen Ausführung ins BASIC zurückzukehren hat.

Das gleiche Problem stellt sich beim GOSUB-Kommando in BASIC, auch dort brauchen Sie ein RETURN, um wieder ins Hauptprogramm zurückzukehren. Um vom Maschinencode ins BASIC zurückzukehren, lautet der entsprechende Maschinencode RET. Er entspricht dem Dezimalcode 201.

Es gibt noch eine weitere Sache, die wir beachten müssen, bevor wir uns dem Schreiben eigener Maschinenroutinen zuwenden. Wenn wir PRINT USR xxx eingeben, wird, nach Ausführung der Routine, die ausgedruckte Zahl dem Wert des Registerpaares BC entsprechen.

Beispiel:

```
LD B,0
LD C,0
RET
```

Wir können den entsprechenden Maschinencode mit den Werten 06, 0, 14, 00, 201, 999 über die Laderoutine auf Seite 195 eingeben und dann über PRINT USR 32000 die Routine ausführen. Der Wert des Registerpaares BC (0) wird, was uns nicht überrascht, ausgegeben, da in der Routine BC mit 0 geladen wurde.

Versuchen Sie nun einmal, den Wert des zweiten und vierten Byte zu verändern, das sind nämlich die Werte, die an B und C übergeben werden. Der Wert für B enthält das Vielfache von 256 und C ist der Rest zwischen 0 und 255. Wenn Sie den Wert ausrechnen, mit dem Sie ein Registerpaar laden wollen (LD HL,xxx xxx), ist es wichtig, daran zu denken, daß zuerst das niederwertige Byte, also der Rest zwischen 0 und 255, und dann erst das Vielfache von 256 übergeben wird. Bei HL entspricht also L dem Low-Byte, dem niederwertigen, und H dem High-Byte, dem höherwertigen. Wenn z.B. das Vielfache von 256 Null ist und ein Rest von 100 besteht, so bedeutet das, daß H=0 und L=100 wird. Zwar schreiben wir als Kürzel LD HL,000 100; der Dezimalcode dafür jedoch lautet 33 100 000. Sobald wir also den Befehl in Dezimalcode übersetzen, vertauschen wir die beiden Zahlen.

Versuchen Sie folgendes:

```
LD B,0      06 00
LD C,0      14 00
LD A,12     62 12
SUB A,6     214 06
LDC,A       79
RET         201
```

Was passiert, ist folgendes:

```
LD B,0     Lade B mit 0
LD C,0     Lade C mit 0
LD A,12    Lade A mit 12
SUB A,6    Subtrahiere 6 von A
LD C,A     Lade C mit dem Wert von A
RET        Rücksprung ins BASIC
```

Im BASIC sähe das wie folgt aus:

```
LET B = 0
LET C = 0
LET A = 12
LET A = A-6
LET C = A
RETURN
```

Die BASIC-Version braucht fast 60 Bytes Speicherplatz, während die entsprechende Maschinenroutine nur 10 Bytes benötigt und zudem noch sehr viel schneller ist.

Das bringt uns schon ans Ende unserer Einführung in den Maschinencode. Sie sehen, es war keineswegs unsere Absicht, aus Ihnen einen Profi in der Maschinencode-Programmierung zu machen, sondern Ihnen vielmehr ein grundsätzliches Verständnis von deren Arbeitsweise zu liefern, damit Sie eine Basis haben, auf der Sie weiter aufbauen können.

Vorschläge zur weiteren Lektüre:

Wadsworth, Nat: Z80 Instruction Handbook, Scelbi Publications, USA, 1978

Dieses Buch erklärt mit leicht verständlichen Ausführungen die Fähigkeiten des Z80-Befehlssatzes. Es dient als Nachschlagwerk für die gängigen Befehlsabkürzungen, den Maschinencode und die Anwendungen der verschiedenen Befehlsarten in der Z80 CPU. Wadsworth sagt zu diesem Buch: „Es ist als praktischer Führer für den Neuling gedacht sowie als Arbeitshilfe für professionelle Programmierer, die im Maschinencode bzw. auf Assemblerebene mit Z80-Mikrocomputern arbeiten müssen."

Wadsworth, Nat: Z80 Software Gourmet Guide and Cookbook, Scelbi Publications, USA, 1979

Dieses Buch gibt eine vollständige Beschreibung des Z80-Befehlssatzes und eine Vielzahl von Programmhilfen in Form von nützlichen Unterprogrammen.

Zaks, Rodnay: Programmierung des Z80, SYBEX Verlag GmbH, Düsseldorf, 1982

Kapitel 10

Ein Leitfaden für besseres Programmieren

Bei der Computerprogrammierung hat sich, wie auf den meisten Gebieten menschlicher Bildung, ein Beurteilungsprinzip entwickelt, welches bestimmte Techniken als gut, nicht so gut oder grundlegend schlecht klassifiziert. In der Anfangszeit wird es für die Arbeit mit Ihrem Spectrum schon einen Wert in sich darstellen, wenn Sie ein lauffähiges Programm erstellt haben, welches mehr oder weniger das macht, was Sie erreichen wollten. Je mehr Zeit Sie jedoch auf die Programmierung verwenden, desto lohnender wird es Ihnen wohl erscheinen, Ihre Technik zu verbessern. Das würde Ihnen nicht nur die Fehlersuche erleichtern, sondern auch die Ausarbeitung des jeweiligen Teils erheblich vereinfachen, um zu dem beabsichtigten Ziel zu kommen. Eine spätere Weiterentwicklung des Programms wird viel einfacher sein, wenn es sauber gegliedert ist, als wenn es aus einer Masse unübersichtlich gehaltener GOTO's und verschachtelter IF THEN's besteht. Ein gut geschriebenes Programm ist also einfach benutzerfreundlicher.

Die Regeln hierfür jedoch sind nicht auf Steintafeln gemeißelt und sollten daher nicht einfach kritiklos akzeptiert werden. So soll z. B. die Verwendung von unbedingten GOTO's von einem schlechten Programmierstil zeugen; sie läßt sich jedoch in einem BASIC, wie es vom Spectrum unterstützt wird, der keine Prozeduren besitzt und kein REPEAT/UNTIL oder DO/WHILE, oft nicht vermeiden. Trotzdem ist es sinnvoll, das Programm auf direkte GOTO's hin zu untersuchen, um zu schauen, ob sie eventuell vermieden werden können, z. B. durch Verwendung einer FOR/NEXT-Schleife, einer Unterroutine oder durch Verschieben eines Blocks an eine andere Stelle im Programm.

Es gibt ein weiteres Beispiel guten Programmierstils, das Sie beim Spectrum getrost vergessen können. So wird es z. B. für schlecht angesehen, aus einer FOR/NEXT-Schleife zu springen, ohne sie ordnungsgemäß zu beenden. Es gibt sogar Computer, wie den Acorn Atom von BBC, bei dem dies zu einem Absturz führen würde. Mit Ihrem Spectrum können

Sie derartiges ungestraft tun, so oft Sie wollen, und eine unvollendete FOR/NEXT-Schleife ist immerhin noch eine kleinere Sünde als IF X=3 THEN LET X=X+1: GO TO Y, was oft die einzige Alternative zu einer vorzeitig beendeten FOR/NEXT-Schleife darstellt.

Denken Sie also daran, wenn Sie den Rest dieses Kapitels lesen, daß die sogenannten Regeln mehr eine Sammlung von Vorschlägen oder Richtlinien darstellen. Also lesen Sie sie ruhig, denken Sie darüber nach, aber nehmen Sie sich die Freiheit, sie zu ignorieren, wenn Sie sie nicht für die beste Idee in Ihrem bestimmten Anwendungsfall halten. Dennoch gab es sicher gute Gründe für die Aufstellung dieser Regeln, so daß Sie wahrscheinlich eher einen Nutzen durch deren Anwendung als durch deren Abweisung ziehen.

Viele Programmierhandbücher schlagen z.B. vor der Programmierung das Erstellen eines Flußdiagramms mit Kästchen und Karos vor, bei denen die Entscheidungen des Computers durch Pfeile dargestellt werden, die den Verlauf des Programms in der Ablaufphase kennzeichnen. Obwohl es nicht nötig ist, ein sorgfältig ausgearbeitetes Flußdiagramm zu erstellen, werden Sie es manchmal doch für ganz sinnvoll erachten, ein grobes Flußdiagramm zu Papier zu bringen, um Ihre Gedanken zu ordnen, anstatt damit anzufangen, einfach Programme in Ihren Computer hineinzuhacken.

Manchmal ist eine Serie von Stichworten, eventuell durch Pfeile miteinander verbunden, das einzige, was Sie brauchen. Für ein Spiel des Typs AUSBRUCH könnte das wie folgt aussehen:

„Variablen definieren, Mauer an der Bildschirmoberkante darstellen, Ball bewegen, Tastatur abfragen (Schlägerbewegung?), testen ob der Ball einen Mauerziegel getroffen hat, wenn ja, Ziegel löschen und Punktezähler aufaddieren, testen ob der Ball schon die Bildschirmunterkante erreicht hat, Vergleichen der Schlägerposition mit der Ballposition, bei Übereinstimmung Ballreflektion Richtung Mauer, sonst Ballzähler um Eins verkleinern und nachschauen, ob noch Bälle übrig sind, wenn nein, ist das Spiel zu Ende, wenn ja, springe zu der Zeile, die die Ballbewegung ausführt."

Wenn Sie wollten, könnten Sie nun anhand dieser Angaben ein AUSBRUCH-Programm schreiben. Ohne diese Stichworte würden Sie vielleicht am Ende des Programms feststellen, daß Sie irgend etwas Wichtiges vergessen haben. Sie müßten dann entweder versuchen, den fehlenden Teil zwischen den existierenden zu quetschen oder ihn mit einem häßlichen GOTO an das Programmende zu hängen oder auf irgendeine Weise darum herumzukommen.

Jedesmal, wenn der Spectrum innerhalb seines Programms an ein GO TO oder ein GO SUB kommt, muß er das ganze Programm von der ersten

Zeile an, Zeile für Zeile durchsuchen, bis er die gewünschte Zeile erreicht hat. Daher wird ein Programm umso langsamer, je weiter hinten in einem Programm er die gesuchte Zeile findet. Zwar ist die dadurch entstehende Verzögerung sehr gering, in einem Programm mit sich bewegender Grafik jedoch kann jede derartige Verzögerung des Programms zu weniger Effizienz führen. Aus diesem Grunde sollten Sie auch oft benutzte Unterroutinen so weit wie möglich an den Programmanfang setzen. Wenn die Geschwindigkeit eines Programms von Bedeutung ist, könnten Sie das Programm z. B. mit einem REM beginnen, dort den Programmnamen eintragen, dann, sagen wir einmal, zu Zeile 9000 springen, wo die Variablen initialisiert werden, und dort könnte die Benutzeranleitung stehen. Dies hat zwei Vorteile. Zum ersten bedeutet es für den Computer, daß er nicht jedesmal, wenn er nach einem GOTO oder GOSUB die betreffende Zeile sucht, die Initialisierungsroutine und Benutzeranleitung durchgehen muß. Zum zweiten: Wenn Sie merken, daß Sie eine benötigte Variable vergessen haben, können Sie diese einfach an das Programm anhängen, und zwar direkt vor das abschließende RETURN. Das ist besser als der Versuch, es am Anfang irgendwo zwischenzuquetschen.

Ein weiteres Ziel, welches Sie sich vor Augen halten sollten, wenn Sie ein Programm schreiben, ist, daß jeder, der es einmal benutzt, in der Lage sein sollte zu verstehen, was da vor sich geht. Das Programm sollte daher entweder klärende Informationen enthalten oder in irgendeiner anderen Art und Weise damit versorgt werden. Eine Kombination zwischen schriftlichen Anweisungen auf Papier mit einer entsprechenden Kurzinformation davon innerhalb des Programms (z. B. welche Taste bei einer Eingabe was bewirkt) ist unter Umständen besser als ein detailliertes Bündel von möglichen Anweisungen innerhalb des Programms. Zwar ist dies nicht so wichtig, wenn das Programm nur für Ihren Privatgebrauch konzipiert ist, jedoch unerläßlich für Programme, die später verkauft oder sonstwie für andere Leute nutzbar gemacht werden sollen.

Es ist ebenfalls wichtig, fehlerhafte Eingaben abzufangen, bevor Sie einen Fehler verursachen. Wie man dies machen kann, ist schon im Kapitel „Der Spectrum als Lehr- und Lernmittel" (S. 81) diskutiert worden. Das sind Techniken, die es dem Anwender erlauben, eine Eingabe, bevor sie endgültig vom Programm übernommen und weiterverarbeitet wird, erst einmal zu untersuchen. Wichtig ist dies besonders bei Programmen, die ein großes Quantum an Eingaben zu verarbeiten haben. Ein nachfolgender Test sowie die Möglichkeit, Eingabeinformationen nachträglich zu verändern, ist auch keine schlechte Idee. Viele Programme des kommerziellen Bereichs beinhalten diese Möglichkeiten.

In Ergänzung zu dem oben Aufgeführten sollten Benutzeranfragen klar verständlich sein. So kennzeichnet ein blinkender Cursor an der Bildunterkante zwar, daß eine Eingabe erwartet wird, aber solange keine Ergänzungsinformation auf dem Bildschirm erscheint oder zusammen mit dem

Eingabebefehl ausgegeben wird, mag der Anwender zweifeln, was er eingeben soll.

Eine separat geführte schriftliche Liste der Eingabeanforderungen ersetzt in keinem Fall Benutzerinformationen innerhalb des Programms. Zwar können viele auf dem ZX Spectrum geschriebene Programme nicht mit dem Luxus ausführlicher Benutzerinformationen aufwarten, ganz einfach wegen der Beschränkung des verfügbaren Speichers, jedoch ist dies keine generell anwendbare Entschuldigung beim Erstellen von Programmen auf dem Spectrum.

REM-Angaben, die erklären, was der folgende Programmteil genau machen soll, stellen eine brauchbare Hilfe dar, um Programmlisten transparent zu halten, und bieten zusätzlich eine Gewähr, daß man auch dann noch weiß, was das Programm macht, wenn man sich erst nach längerer Zeit wieder daran setzt. Wenn Sie dies noch lernen wollen, schauen Sie sich das Programm KATZEN UND ANDERE DINGE an. Dort sehen Sie den ordnungsgemäßen Gebrauch von REM-Angaben. Allein wenn Sie sich die REM-Zeilen anschauen, reicht diese Information aus für alles, was Sie über das Programm wissen müssen. Sie könnten sogar, wenn Sie wollten, das ganze Programm anhand der REM-Zeilen rekonstruieren oder neu schreiben: Auswahl und Konstruktion eines Objekts; Darstellung des Objekts; Korrekturen der Eingabe; lobende Worte; Eingabe der einzelnen Tasten; DATA-Zeilen. Es dürfte somit wohl außer Zweifel stehen, daß ein Programm, welches Informationen in Form von REMs enthält, leichter zu entwirren ist als eines ohne.

Nachdem Sie Ihr Programm geplant und in Hauptgruppen unterteilt haben, sollten Sie noch weiter gehen und auch die jeweiligen Untergruppen beschreiben. Dadurch, daß Sie jede Routine in kleinere Unterroutinen zerlegen, können Sie feststellen, daß das Programm sich praktisch selbst schreibt. Ihnen werden dann auch mögliche Fehler und Unterlassungssünden auffallen, und dies möglicherweise, bevor Sie auch nur eine einzige Zeile eingetippt haben! Es ist ebenso nützlich, nebenher auf einem separaten Stück Papier alle benutzten Variablen zu notieren. Wenn Sie dann nämlich den Programmteil erreichen, in dem den Variablen Werte zugewiesen werden, haben Sie schon eine komplette Liste zur Hand.

Es wird Ihnen auch erheblich die Arbeit erleichtern, wenn Sie für die Variablen, die Sie benutzen, Namen verwenden, die deren Zweck entsprechen, wie PUNKTE für die erreichten Punkte im Spiel etc. Obwohl dies ein wenig mehr Eingabezeit benötigt als die Verwendung einer einstelligen Variablen, wird dadurch ganz nebenbei die Wahrscheinlichkeit verringert, dieselbe Variable innerhalb eines langen Programms zweimal für verschiedene Funktionen zu verwenden, was zu Programmabbrüchen führen kann, deren Ursache kaum festzustellen ist. Je länger Variablennamen also sind, desto mehr Speicherplatz nehmen sie zwar in Anspruch,

aber bei den Programmen, die Sie mit Ihrem Spectrum erstellen, fällt dies kaum ins Gewicht. Auch die Ersteller kommerzieller Software haben längst festgestellt, wie sinnvoll die Verwendung derart expliziter Variablennamen sein kann. In unserem Finanzprogramm z. B. verwendeten wir Variablennamen wie SUMME, DURCHSCHNITT und ANZAHL.

Keiner verlangt von uns, das Rad neu zu erfinden. Trotzdem ist es nicht gerade die feinste Art, Programme von anderen zu stehlen, sich anzupassen und dann als Eigenentwicklung weiterzuleiten. Eine Sortierroutine in stundenlanger Programmierarbeit auszuarbeiten hat z. B. wenig Zweck, wo eine große Anzahl passender Routinen durch Veröffentlichung schon zur Verfügung steht. Es ist nicht sinnvoll, eine Routine neu herzustellen, die es bereits gibt, oder die mit geringen Anpassungen aus einer vorhandenen abgeleitet werden könnte, zumal es bei gewissen Aufgabenstellungen sowieso nur eine begrenzte Anzahl von Lösungsansätzen gibt. Das trifft allerdings nur zu, wenn man bezüglich der Fertigstellung eines bestimmten Programms unter Zeitdruck steht. Immerhin wird die Genugtuung, eine Routine von Grund auf selbst erstellt zu haben, egal welchen Standard sie hat, und der größere Einblick in die Materie Sie mehr als entschädigen.

Viele Programmierer werden ganz schnell zum Profi im Gebrauch bestimmter Teile des Spectrum BASIC, machen aber dann den Fehler, das Handbuch aus der Hand zu legen und nicht mehr dieselbe Energie an die Erforschung der übrigen, im BASIC verfügbaren Kommandos zu verwenden. Unabhängig davon, wie vertraut Sie sich bei der Arbeit mit dem Spectrum und dessen BASIC-Dialekt fühlen, sollten Sie doch von Zeit zu Zeit das Manual (und auch andere Computerliteratur) zur Hand nehmen, einfach um einmal nachzuschauen, ob es da noch etwas gibt, was Sie mißverstanden haben oder noch nicht kennen.

Möglicherweise ist es Ihnen auch eine Hilfe, im Geiste einmal bestimmte Teile des Programms zu „starten", so als ob Sie selbst der Computer wären. Beginnen Sie am Anfang des Programms, gehen Sie Anweisung für Anweisung durch. Wenn Sie derart ein Programm zu analysieren versuchen, werden Sie oft auf schlampig geschriebene Routinen stoßen oder solche, die sich als potentielle, unsichtbare Fehlerquellen entpuppen. Die „geistige Ablaufsimulation" eines Programms ist ebenso ein guter Weg festzustellen, ob an irgendeiner Stelle des Programms der Computer z. B. Division durch 0 durchzuführen versucht, was eine Fehlermeldung und den Programmabbruch verursachen würde.

Sorgen Sie dafür, daß die Ausgabeinformationen auf dem Bildschirm für den Anwender klar verständlich sind. Sie können kein Programm schreiben und davon ausgehen, daß Sie jedesmal, wenn Sie es laufen lassen, dem Anwender über die Schulter schauen! Ein Programm, zu dem Sie Erläuterungen geben müssen wie „Oben rechts in der Ecke steht die Anzahl

der Versuche, die Sie bisher schon ohne Erfolg für die Antwort gebraucht haben" oder „Die erste Zahl, die auf dem Bildschirm erscheint, stellt das Ergebnis der Jahreskumulation und die zweite die avisierten Verkäufe für das nächste Quartal dar" könnte man kaum als gut bezeichnen.

Die in diesem Kapitel gemachten Vorschläge fallen in zwei Kategorien:

1. Erst überlegen und dann programmieren.
2. Testen Sie die Wirkung Ihres Programms (hinsichtlich der Eingabe und der angegebenen Resultate) mit einem Benutzer, der das Programm nicht geschrieben hat.

Wenn Sie Ihr Programm unter Beachtung dieser beiden Punkte schreiben, werden Sie sehen, wie schnell sich Ihr „Programmierstil" wesentlich verbessert.

Vorschläge zur weiteren Lektüre:

Ledin, George und Ledin, Victor: The Programmer's Book of Rules, Lifetime Learning Publications, USA, 1979

Savage, Earl R.: BASIC Programmer's Notebook, Howard Sams & Co., Inc., USA, 1982

Nagin, Paul A. und Ledgard, Henry F.: BASIC With Style, Hayden Book Company, Inc., USA, 1978

Meek, Brain und Heath, Patricia (Hrsg.): Guide to Good Programming Practice, Ellis Horwood Ltd./John Wiley & Sons, 1980

Anhang A

Geschichtlicher Überblick

Die älteste Rechenmaschine, die man als solche bezeichnen kann, ist wahrscheinlich der „Abacus", ein viereckiger Holzrahmen, in dessen Innern sich parallele Verbindungsstäbchen mit aufgereihten Perlen befinden. Die Position der Perlen auf dem Stab kennzeichnet einen bestimmten Zahlenwert. Später dann wurde der Abacus zuerst durch den Rechenschieber, dann durch den Taschenrechner ersetzt.

John Napier aus Schottland ist einer der ersten, die einen wesentlichen Beitrag zur Entwicklung einer rechnenden Maschine geleistet haben. Napier erfand, was später unter dem Namen „Napier's Rechenstäbe" bekannt wurde, neun viereckige, unterteilte Stäbe, die die Multiplikation durch Addition der auf den Stäben dargestellten Zahlen durchführten. Der englische Geistliche William Oughtred entwickelte schon 1621 ein Gerät, mit dessen Hilfe er über Logarithmen Zahlen multiplizierte.

Später dann, im 17. Jahrhundert, war es Blaise Pascal, der mit Hilfe von ineinandergreifenden Zahnrädern eine Methode für die Addition von Zahlen entwickelte. Die Zahnräder befanden sich in einem Gehäuse mit kleinen Fenstern, in denen das Ergebnis einer derartigen Addition dargestellt wurde. Gottfried Leibniz schließlich, ein Zeitgenosse Pascals, fand einen Weg, Multiplikation zu mechanisieren. Diese Methode war derart exzellent, daß sie noch 250 Jahre später bei Rechenmaschinen verwendet wurde.

Im Jahre 1792 wurde Charles Babbage in Devon geboren. Er hat eine große Bedeutung in der Computergeschichte. Man kann vielleicht sogar behaupten, daß er den allerersten Computer erfand. Charles Babbage war unzufrieden mit der Ungenauigkeit der damals verfügbaren mathematischen Tabellenwerke und versuchte daraufhin, mit Hilfe einer von ihm entwickelten Maschine genauere Tabellen zu erstellen. Dies scheiterte nur an der Unfähigkeit damaliger Ingenieure, Teile der Maschine mit der geforderten Genauigkeit zu bauen. Babbage verlor daraufhin das Interesse an seiner Entwicklung und wandte sich der Konstruktion einer „analy-

tischen Maschine" zu, die, wäre sie fertiggestellt worden, wohl der erste existierende Computer gewesen wäre, denn diese Maschine war dazu ausgelegt, nicht nur die Grundrechenarten durchzuführen, sondern auch anhand eines erlangten Ergebnisses Entscheidungen zu treffen.

Der mathematische Hintergrund, nach dessen Prinzip der Computer arbeitet, wird Boolsche Algebra genannt. Von dem Mathematiker und Logiker George Boole entwickelt, und zwar lange bevor der erste Computer das Licht der Welt erblickte, war er der erste, der gelochte Karten für die Aufzeichnung und Erfassung einer amerikanischen Volkszählung verwendete. Das Lochkartenprinzip war von Hermann Hollerith entwickelt worden, der eine Vertriebsgesellschaft für dieses System gegründet hatte. Die Gesellschaft florierte, kaufte Konkurrenten auf und wurde schließlich in „International Business Machine Corporation" umbenannt, heute die größte Computerfirma der Welt: IBM.

1870 entwickelte der britische Physiker Lord Kelvin ein Gerät, um Gezeiten vorherzusagen, und machte dann später den Vorschlag, eine Maschine, die er die „Maschine zur Differentialanalyse" nannte, zu bauen und nicht nur für Vorhersagen, sondern generell zur Lösung von Differentialgleichungen zu verwenden. Eine solche Maschine wurde dann auch tatsächlich von einem Professor am technologischen Institut in Massachusetts, Vannevar Bush, gebaut. Obwohl diese Maschine einwandfrei funktionierte, merkte Bush, daß es keine Zukunft für mechanisch arbeitende Rechenmaschinen gab, und ersetzte Teile der Maschine durch Röhren. Ein weiterer Schritt zum modernen Computer war getan.

1940 war es dann George Stibitz, der im Auftrag einer Telefonfirma herausfand, daß Binärinformationen, realisiert durch Schalter mit „ein" zur Darstellung einer 1 und „aus" zur Darstellung einer 0, Informationen speichern konnten. Ein Gerät aus Telefonrelais und kleinen Lämpchen war der erste elektronische Rechner, der Binärarithmetik verarbeitete. Heute arbeiten nahezu alle Computer mit Binärarithmetik, die dann in für uns verständliche Symbole, bestehend aus Ziffern und Buchstaben, übersetzt wird.

Trotzdem dauerte es noch weitere vier Jahre, bis diese Idee sich durchsetzte. In Cambridge, Massachusetts, wurde die erste vollautomatische Rechenmaschine, genannt „Automatic Sequence Controlled Calculator", von Howard Aiken von der Harvard University vollendet, der damals für IBM arbeitete.

Die Computer der ersten Generation waren riesige Maschinen, benötigten wegen ihrer Röhren Unmengen an Strom und waren berüchtigt wegen ihrer Unzuverlässigkeit. Zwar sorgte die Erfindung des Transistors dafür, daß sich die Größe der Computer reduzierte, aber erst durch die Erfindung des Mikroprozessors 1971 wurde es möglich, Computer wie den Spectrum zu realisieren.

Die Idee, einen integrierten Baustein zu schaffen, den Vorläufer des Chip, kam von G. W. Dummer, der für die Firma British Royal Radar arbeitete. Damals nahm kaum einer von seiner Idee Notiz, und erst sechs Jahre später war es Jack Kilby von Texas Instruments, der sie realisierte.

Der erste „richtige" Mikroprozessor kam von der amerikanischen Firma Intel und wurde im November 1971 unter dem Namen 4004 vorgestellt. Sowohl Intel als auch der Rest der Welt erkannten nicht, welche enormen sozialen Veränderungen ihr Produkt hervorrufen würde. Fast ein Jahr verging, bis man merkte, daß der Grundstein für eine technologische Revolution bereits gelegt war.

Der erste Personal Computer, der Altair, wurde 1975 von einer kleinen Firma Mits in Neu-Mexiko vorgestellt. Das Rennen, immer kleinere, leistungsfähigere Computer zu entwickeln, hatte begonnen.

Clive Sinclair gründete seine erste Firma, Sinclair Radionics, im Jahre 1962 und stellte dort Radios und Verstärkerbausätze für den Versandhandel her. Um 1967 lag der Jahresumsatz bei etwa 400000,– DM, und das Sortiment erweiterte sich auf HIFI-Geräte. Fünf Jahre später stieg Sinclair mit dem Executive in den Taschenrechnermarkt ein, dem ersten Taschenrechner, der weltweit Schlagzeilen machte. Die Welt horchte erneut auf, als im Januar 1977 der Welt erster „Taschen"-Fernseher Microvision mit einer 5-cm-Bildröhre erschien. Zwei Jahre später schon wurde diese Version in Großbritannien zur Hälfte des ursprünglichen Preises verkauft.

Ende des Jahres 1980 brachte Clive Sinclair seinen ersten Computer, den ZX80 heraus. Er war zu dieser Zeit der billigste Computer auf dem Markt und hat wohl eine entscheidende Rolle bei dem Preisverfall von Kleincomputern auf dem Markt gespielt. So wurde Sinclair fast über Nacht zum größten Computerhersteller in Großbritannien. Mit dem Nachfolger des ZX80, dem ZX81, gelangte Sinclair dann zu Weltruhm.

Der ZX Spectrum, der im April 1982 in Großbritannien auf den Markt kam, ist wiederum ein würdiger Nachfolger des ZX81, da er neben den Fähigkeiten des ZX81 mit größerer Geschwindigkeit, Farbe und hochauflösender Grafik aufwarten kann. In seinem Gefolge kam es zu einem Preiseinbruch bei anderen Kleincomputern, und schon bald wurde eine Vielzahl weiterer Kleincomputer mit Farbdarstellung angekündigt.

Vorschläge zur weiteren Lektüre:

Evans, Christopher: The Making of the Micro, Gollancz, 1982

Lavington, Simon: Early British Computers, Manchester University Press, 1980

Hartnell, Tim: The Personal Computer Guide, Virgin Books, 1982

Buchsbaum, Walter H.: Personal Computers Handbook, Howard W. Sams & Co., Inc., USA, 1981

Bradbeer, Robin: The Personal Computer Book, Glower Publishing Co., 1982

Ditlea, Steve: A Simple Guide to Home Computers, A&W Visual Library, USA, 1979

Solomon, Leslie und Veit, Stanley: Getting Involved With Your Own Computers, Enslow Publishers, USA, 1977

Bunnell, David: Personal Computing – a Beginner's Guide, Dutton, USA, 1978

Anhang B

Peripheriegeräte

So nennt man im Computerjargon Geräte, die zwar mit einem Computer zu tun haben, jedoch kein direkter Bestandteil davon sind. Zu den wichtigsten Peripheriegeräten zählt man für die Eingabe zuständige Peripherie (z. B. Tastatur, heutzutage meist im Rechner eingebaut), für die Ausgabe zuständige Peripherie (Bildschirm, Drucker) sowie Massenspeicherperipherie (Disk- oder Bandstation). Die meisten Großrechner besitzen Nebenstellencomputer, welche für sie die Verwaltung der Peripherie übernehmen. Auch diese Nebenstellencomputer sind erheblich größer als der Spectrum.

Für ihn gibt es, was derartige Peripherie angeht, eine Interfacekarte, genannt RS232, die es erlaubt, eine große Menge Peripherie wie z. B. Drukker, Terminals oder sonstige Peripherie anzuschließen. Die RS232 ist eine genormte Standardschnittstelle, für die inzwischen eine Vielzahl von Anschlußgeräten angeboten wird.

Drucker

Ein Drucker wird für die meisten Anwendungen benötigt. Man unterscheidet verschiedene Typen:

Lineprinter: ein großes, teures Gerät, welches eine ganze Zeile auf einmal drucken kann, normalerweise in einer Geschwindigkeit von bis zu 600 Zeilen à 132 Zeichen pro Minute.

Daisywheel: Typenraddrucker, die mit Anschlag drucken, wie bei einer Schreibmaschine.

Dotmatrix: Zeichen werden elektrostatisch, thermisch oder per Anschlag als Punktfolge erstellt.

Impact: Drucker mit Anschlag und Farbband.

Thermo: Papier, das durch Erhitzen geschwärzt wird. Diese Drucker können sehr schnell arbeiten.

Elektrostatisch: aluminiumbeschichtetes Papier, das sich, wenn ein vom Drucker erzeugter Funke auftrifft, an dieser Stelle schwarz färbt.

Was man in der Regel von einem Drucker erwartet, ist Geschwindigkeit und ein gutes Druckbild. Papier für elektrostatisch und thermisch arbeitende Drucker ist zwar teuer, dafür sind allerdings die Drucker entsprechend billiger. Wenn Ihnen nur daran liegt, Informationen preisgünstig (und nicht in Riesenmengen) auf Papier zu ziehen, reicht der von Sinclair angebotene Drucker vollkommen aus. Die untere Preisgrenze bei elektrostatischen Druckern liegt bei ca. 240 DM (Sinclair-Drucker mit 32 Zeichen/Zeile), Matrixdrucker kosten ab ca. 800 DM, solche auf Thermopapierbasis etwas weniger. Daisywheeldrucker bekommt man ab ca. 1800 DM, schnelle kosten bis zu 12000 DM.

Datenträger

Die meisten Datenträger arbeiten auf der Basis von Magnetband oder magnetischen Scheiben. Für die Aufzeichnung von Spectrum-Programmen wird sich wohl ein simpler Monorecorder als optimal herausstellen. Vermeiden Sie möglichst die Verwendung eines Stereogeräts. Wenn möglich, benutzen Sie eine Maschine, die für gute Aufzeichnung digitaler Informationen bekannt ist (das braucht nicht unbedingt ein teures Gerät zu sein). Vermeiden Sie das Arbeiten auf Batteriestrom, verwenden Sie Netzstrom. Achten Sie auf einen Recorder mit Zählwerk sowie manueller Aussteuerungsmöglichkeit bei Aufnahme. Ebenfalls sinnvoll ist eine Aussteuerungsanzeige sowie Funktionen, die eine nachträgliche Bandkontrolle ermöglichen. Eine Elektronik, die Tonsignale filtert, ist nicht wünschenswert. Falls Ihr Gerät einen derartigen Filter besitzt, schalten Sie ihn bei der Aufzeichnung ab oder auf Minimalposition.

Verwenden Sie nur gutes Bandmaterial, welches nie länger als C60 sein sollte. TDK-Cassetten vom Typ D sind ausgezeichnet. Firmen wie BASF und AGFA produzieren ebenfalls zufriedenstellendes Bandmaterial. Speziell für Computer entwickelte C12- und C15-Bänder sind von der Bandlänge her die besten, obwohl nicht alle Bänder ausreichende Qualität besitzen.

Diskettenlaufwerke sind sehr viel schneller als Bandlaufwerke. Sie gewährleisten, daß Informationen irgendwo auf der Platte schnell und ohne Zurückspulen gefunden werden können. Sie können sogar von mehreren Datenfeldern in schneller Folge parallel lesen oder, was ebenfalls möglich ist, verschiedene Operationen auf derselben Diskette vornehmen, z. B. Lesen von Daten aus einer Datei, Zurückschreiben der modifizierten Daten auf dasselbe Datenfeld oder ein anderes. Dies kann auf dem gleichen

Laufwerk erfolgen. Auf einem Band wäre dies unmöglich, es würde ständiges, präzises Hin- und Zurückspulen nötig machen.

Das Mikrodiskettenlaufwerk, eine spezielle Form des Diskettenlaufwerks, kann bis zu 100 K an Informationen speichern. Bis zu acht Laufwerke können an den Spectrum angeschlossen werden. Die Übertragungsgeschwindigkeit der Daten beträgt 16 K pro Sekunde, während es beim Recorder nur 1.5 K sind.

Die Speicherung großer Datenmengen ist nie 100% verläßlich; es ist daher immer sinnvoll, Kopien der wertvollen Informationen anzufertigen. Zwar kann bis zu einem gewissen Grad eine Sicherung auf Drucker ausreichen, doch wird Ihr Vertrauen sehr schnell schwinden, wenn Sie einmal die Arbeit mehrerer Monate vom Papier wieder in den Rechner tippen müssen. Sicherheitskopien brauchen nicht auf demselben Datenträgertyp wie das Original gemacht zu werden; die Sicherung eines Diskette-ninhalts kann ohne weiteres auf Band erfolgen. Trennen Sie jedoch sorgfältig die Disketten, mit denen Sie immer arbeiten, von den angefertigten Sicherungskopien, denn oft kann eine gespeicherte Information mehr wert sein als der Computer selbst. Halten Sie Disketten von Magnetfeldern fern, wie sie bei laufenden Motoren entstehen. Löschen Sie nie eine Diskette, bevor Sie sicher sind, daß Sie auch tatsächlich die richtige löschen.

Anhang C

Der Spectrum in Stichworten

Abmessungen

Länge: 233 mm
Breite: 144 mm
Höhe: 30 mm

CPU/Speicher

Z80A Microprozessor mit 3.5 MHz Taktfrequenz. 16 K ROM mit BASIC-Interpreter und Betriebssystem. Wahlweise 16 oder 48 K RAM.

Tastatur

40 Tasten mit Groß-/Kleinschreibung und Dauerfunktion für Großbuchstaben. Alle BASIC-Schlüsselwörter über eine Taste erreichbar, zusätzlich 16 Funktionstasten für Grafiksteuerung, 22 Farb-/Steuerzeichen und 21 vom Benutzer definierbare Zeichen. Alle Tasten haben auch Dauerfunktion.

Darstellung

Speichergepufferte Darstellung von 256 x 192 Bildpunkten sowie pro Bildschirmquadrat je ein zusätzliches Attributbyte, wo eine der acht Vorder- und Hintergrundfarben sowie die Darstellungsattribute „normale Helligkeit", „verstärkte Helligkeit", blinkende oder ständige Darstellung erfaßt sind. Anschließbar sind Farbfernseher nach dem PAL-System auf Kanal 36.

Ton

Interner Lautsprecher, der vom Rechner aus zur Tonausgabe in über mehr als zehn Oktaven über das BASIC-Kommando BEEP angesprochen werden kann. Buchsen an der Computerrückseite erlauben den Anschluß eines externen Verstärkers.

Grafik

Befehle zur Punkt-, Linien-, Kreis- und Bogendarstellung in Hochauflösung. 16 vordefinierte und 21 vom Anwender definierbare Zeichen. Ebenso Funktionen zum Festhalten eines Zeichens an einer bestimmten Bildschirmposition, zum Feststellen der Darstellungsart und der gesetzten Bildpunkte.

Zum Schreiben von Text stehen 24 Zeilen à 32 Zeichen auf dem Bildschirm zur Verfügung. Text und Grafik können beliebig gemischt werden.

Farben

Das Setzen der Vorder- und Hintergrundfarben, Helligkeit der Darstellung und Blinken wird durch die BASIC-Befehle INK, PAPER, BRIGHT und FLASH spezifiziert. Zusätzlich kann noch OVER verwendet werden, wodurch eine Überlagerung bei Bildschirmdarstellung aktiviert wird; sie bezieht sich auf das Überschreiben schon bestehenden Textes auf dem Bildschirm. INVERSE bewirkt eine inverse Darstellung des Bildschirminhalts. Diese 16 Kommandos können global gesetzt werden und beziehen sich dann auf PRINT-, PLOT-, DRAW- oder CIRCLE-Kommandos. Sie können jedoch auch innerhalb dieser Kommandos verwendet werden und wirken sich dann nur auf deren Ergebnis aus. Eine weitere Verwendung findet sich bei dem Kommando INPUT, wo der begleitend ausgegebene Text mit derartigen Attributen versehen werden kann. Farbsteuerzeichen, die von der Tastatur eingegeben werden, können in Textsegmente implementiert werden, wo sie genauso ausgeführt werden. Mit Helligkeits- und Blinksteuerzeichen kann analog verfahren werden. Farbsteuerzeichen haben beim Programmlisting keinen Einfluß auf deren Ausführung. Die Bildschirmrandfarbe wird durch das Kommando BORDER gesetzt. Die acht verfügbaren Farben sind schwarz, blau, rot, violett, grün, türkis, gelb und weiß. Alle acht Farben können gleichzeitig auf dem Bildschirm dargestellt werden, wobei manche blinken und manche heller dargestellt werden können.

Bildschirm

Der Bildschirm ist in zwei Sektionen unterteilt. Die obere, normalerweise die ersten 22 Zeilen, stellt die Programmliste oder in der Ausführungsphase das Ergebnis des Programms dar. Die untere Sektion, normalerweise die untersten beiden Zeilen, zeigt die gerade eben eingegebene Zeile oder das gerade eingegebene Kommando, oder stellt die gerade editierte Zeile dar. In dieser Zeile sind auch die Editierfunktionen mit Hilfe der Cursorsteuerzeichen sowie Einfügen und Löschen verfügbar. Die untere Bildschirmsektion kann sich ausbreiten, um Programmzeilen bis zu 22 Bildschirmzeilen Länge zu erfassen.

Mathematische Operationen und Funktionen

Arithmetische Operationen sind +, −, ∗, / und Exponentiation. Mathematische Funktionen sind Sinus, Cosinus, Tangens und deren Arcus, Natürlicher Logarithmus, Exponentialfunktion, Signumfunktion, Absolutwertfunktion, Rundungsfunktion, Quadratwurzel, Zufallszahlengenerator und Pi.

Zahlen werden als fünf Byte lange Gleitpunktzahlen binär erfaßt, wodurch sich ein Bereich von +/−3∗10 hoch −39 bis +/−7∗10 hoch 38 mit neuneinhalbstelliger Genauigkeit ergibt. Dualzahlen können direkt über die BIN-Funktion eingegeben werden. =, >, <, >=, <= sowie <> können sowohl zum Vergleich von Zeichenketten als auch zum Vergleich von arithmetischen Werten benutzt werden. Außerdem besteht die Möglichkeit, sie in logischen Ausdrücken zu verwenden (0 = logisch falsch, 1 = logisch wahr). Die logischen Operatoren AND, OR und NOT führen logische Verknüpfungen durch und ergeben entweder 0 für logisch falsch oder 1 für logisch wahr.

Vom Benutzer sind Funktionen über DEF FN definierbar, und zwar bis zu 26 numerischen und 26 Zeichenkettenfunktionen.

Daten können über DATA erfaßt und über READ gelesen werden. Der DATA-Pointer kann durch den Befehl RESTORE zurückgesetzt werden.

Eine „Echtzeituhr" ist erhältlich.

Zeichenketten-Operationen und -Funktionen

Zeichenketten können mit Hilfe von + verknüpft werden. Zeichenkettenvariablen können mit Hilfe von =, >, <, >= , <= oder <> verglichen werden. Zeichenkettenfunktionen sind VAL, VAL$, STR$ und LEN. Die Funktionen CHR$ und CODE geben zu einer Zahl das entsprechende ASCII-Zeichen bzw. umgekehrt. Ein Zeichenketten-Abruf-Mechanismus besteht in der Form A$(x TO y).

Variablennamen

Ein Variablenname beginnt mit einem Buchstaben, wobei nicht zwischen groß und klein unterschieden wird.

Zeichenketten: A$ bis Z$
Schleifen: A bis Z
Numerische Felder: A bis Z
Zeichenketten-Felder: A$ bis Z$

Felder

Felder können mehrdimensional sein, wobei der Index mit 1 beginnt. Zeichenketten sind im Prinzip Felder, bestehend aus einer bestimmten Anzahl von Zeichen. Zeichenketten-Felder sind dann praktisch Felder von Feldern. Als Besonderheit kann hierbei der letzte Index wegfallen. Das Ergebnis ist dann ein String.

Berechnung von arithmetischen Ausdrücken bei der Eingabe

Immer wenn ein mathematischer Ausdruck, eine Konstante oder eine Variable während der Programmausführungsphase angetroffen wird, tritt eine Routine in Aktion, die diesen Ausdruck untersucht. Es ist also auch erlaubt, als Sprungadressen arithmetische Ausdrücke zu verwenden. Eine weitere Verwendung besteht darin, den Spectrum einfach als Taschenrechner zu benutzen.

Cassetteninterface

Vor dem eigentlichen Abspeichern der Information sendet der Rechner ein Tonsignal als Vorspann. Dies dient dazu, die Anlaufzeit des Recorders zu überbrücken und Schwankungen in der Aufnahmestärke, die besonders bei Recordern mit automatischer Aufnahmepegeleinstellung festzustellen sind, zu überbrücken. In Ergänzung dazu verwendet man einen Schmitt-Trigger zur Beseitigung von Nebengeräuschen bei wiederholter Aufnahme. Im Vorspann werden zudem Informationen wie Programmtyp, -titel, -länge sowie Adreßinformationen erfaßt. Neben dem Programm können komplette Bildschirminhalte, Speicherblöcke, Zeichenketten und Zeichenfelder abgespeichert werden. Speicherblöcke, Programme und Felder können nachträglich verglichen werden. Programme und Felder können auch hintereinandergelegt werden, um sie mit den im Speicher liegenden Informationen zu verketten. Wo zwei gleiche Zeilennummern oder Variablennamen sich überlappen, wird die alte Information überschrieben. Programme können auch derart abgespeichert werden, daß das Programm nach dem Ladevorgang automatisch mit einer bestimmten Zeilennummer startet. Das Cassetteninterface arbeitet mit 1500 Baud und hat zwei 3,5-mm-Cinchstecker.

Erweiterungsanschluß

Dieser beinhaltet sämtliche Daten-, Adreß- und Steuerleitungen vom Z80. Er wird verwendet für den Drucker, die serielle Schnittstelle mit Interface sowie für die Laufwerke. Über die BASIC-Befehle IN und OUT kann man ähnlich wie mit PEEK und POKE Informationen an diese I/O-Ports (Input/Output-Verbindungen) schicken.

ZX81-Kompatibilität

Das BASIC des ZX81 ist im wesentlichen Teil des Spectrum BASIC. Die Unterschiede sind hauptsächlich folgende:

FAST und SLOW: Der ZX Spectrum arbeitet automatisch im Fast Modus, solange er rechnet. Bei der Bildschirmausgabe wird automatisch auf den Slow Modus umgeschaltet.

SCROLL: Immer, wenn der Bildschirm vollgeschrieben ist, fragt der ZX Spectrum über „scroll?", ob er eine Seite weiterblättern soll.

UNPLOT: Der ZX Spectrum kann einen gesetzten Bildpunkt mit Hilfe des Befehls PLOT OVER löschen und erreicht einen UNPLOT.

ZEICHENSATZ: Der ZX Spectrum benutzt im Gegensatz zum ZX81 den ASCII-Zeichensatz.

Die SYBEX Bibliothek

MEIN ERSTER COMPUTER (2. überarbeitete Ausgabe)
von Rodnay Zaks – eine Einführung für alle, die den Kauf oder die Nutzung eines Mikrocomputers erwägen. 305 Seiten, 150 Abbildungen, Format DIN A5, Ref.Nr.: **3020** (1982)

CP/M HANDBUCH MIT MP/M
von Rodnay Zaks – ein umfassendes Lehr- und Nachschlagewerk für CP/M, das Standard-Betriebssystem für Mikrocomputer. 310 Seiten, 100 Abbildungen, Format DIN A5, Ref.Nr. **3002** (1981)

MIKROPROZESSOR INTERFACE TECHNIKEN (3. überarbeitete Ausgabe)
von Rodnay Zaks/Austin Lesea – Hardware- und Software-Verbindungstechniken samt Digital/Analog-Wandler, Peripheriegeräte, Standard-Busse und Fehlersuchtechniken. 435 Seiten, 400 Abbildungen, Format DIN A5, Ref.Nr.: **3012** (1982)

PROGRAMMIERUNG DES 6502 (2. überarbeitete Ausgabe)
von Rodnay Zaks – Programmierung in Maschinensprache mit dem Mikroprozessor 6502, von den Grundkonzepten bis hin zu fortgeschrittenen Informationsstrukturen. 350 Seiten, 160 Abbildungen, Format DIN A5, Ref.Nr.: **3011** (1982)

PROGRAMMIERUNG DES Z80
von Rodnay Zaks – ein kompletter Lehrgang in der Programmierung des Z80 Mikroprozessors und eine gründliche Einführung in die Maschinensprache. 630 Seiten, 200 Abbildungen, Format DIN A5, Ref.Nr.: **3006** (1982)

EINFÜHRUNG IN PASCAL UND UCSD/PASCAL
von Rodnay Zaks – das Buch für jeden, der die Programmiersprache PASCAL lernen möchte. Vorkenntnisse in Computerprogrammierung werden nicht vorausgesetzt. Eine schrittweise Einführung mit vielen Übungen und Beispielen. 540 Seiten, 130 Abbildungen, Ref.Nr.: **3004** (1981)

DAS PASCAL HANDBUCH
von Jacques Tiberghien – ein Wörterbuch mit jeder Pascal-Anweisung und jedem Symbol, reservierten Wort, Bezeichner und Operator, für beinahe alle bekannten Pascal-Versionen. 510 Seiten, 270 Abbildungen, Format 23 x 18 cm, Ref.Nr.: **3005** (1982)

PASCAL PROGRAMME FÜR WISSENSCHAFTLER UND INGENIEURE
von Alan Miller – eine Sammlung von 60 der wichtigsten wissenschaftlichen Algorithmen samt Programmauflistung und Musterdurchlauf. Ein wichtiges Hilfsmittel für Pascal Benutzer mit technischen Anwendungen. 376 Seiten, 120 Abbildungen, Format 23 x 18 cm, Ref.Nr.: **3007** (1982)

POCKET MIKROCOMPUTER LEXIKON
– die schnelle Informations-Börse! 1300 Definitionen, Kurzformeln, technische Daten, Lieferanten-Adressen, ein englisch-deutsches und französisch-deutsches Wörterbuch. 176 Seiten, Format DIN A6, Ref.Nr. **3008** (1982)

BASIC COMPUTER SPIELE/Band 1.
herausgegeben von David H. Ahl – die besten Mikrocomputerspiele aus der Zeitschrift „Creative Computing" in deutscher Fassung mit Probelauf und Programmlisting. 208 Seiten, 59 Abbildungen, Ref.Nr. **3009**

BASIC COMPUTER SPIELE/Band 2
herausgegeben von David H. Ahl – 84 weitere Mikrocomputerspiele aus „Creative Computing". Alle in Microsoft-BASIC geschrieben mit Listing und Probelauf. 224 Seiten, 61 Abbildungen, Ref.Nr.: **3010**

BASIC PROGRAMME – MATHEMATIK, INFORMATIK, STATISTIK
von Alan Miller – eine Bibliothek von Programmen zu den wichtigsten Problemlösungen mit numerischen Verfahren, alle in BASIC geschrieben, mit Musterlauf und Programmlisting. 352 Seiten, 120 Abbildungen, Ref.Nr.: **3015** (1983)

EINFÜHRUNG IN DIE TEXTVERARBEITUNG
von Hal Glatzer – woraus eine Textverarbeitungsanlage besteht, wie man sie nutzen kann und wozu sie fähig ist. Beispiele verschiedener Anwendungen und Kriterien für den Kauf eines Systems. 208 Seiten, 67 Abbildungen, Ref.Nr. **3018** (1983)

EINFÜHRUNG IN WORDSTAR
von Arthur Naiman – eine klar gegliederte Einführung, die aufzeigt, wie WORDSTAR funktioniert, was man damit tun kann und wie es eingesetzt wird. 248 Seiten, 30 Abbildungen, Ref.Nr.: **3019** (1983)

6502 ANWENDUNGEN
von Rodnay Zaks – das Eingabe-/Ausgabe-Buch für Ihren 6502-Microprozessor. Stellt die meistgenutzten Programme und die dafür notwendigen Hardware-Komponenten vor. 288 Seiten, 205 Abbildungen, Ref.Nr.: **3014** (1983)

BASIC ÜBUNGEN FÜR DEN APPLE
von J.-P. Lamoitier – das Buch für APPLE-Nutzer, die einen schnellen Zugang zur Programmierung in BASIC suchen. Abgestufte Übungen mit zunehmendem Schwierigkeitsgrad. 240 Seiten, 185 Abbildungen, Ref.Nr.: **3016** (1983)

CHIP UND SYSTEM: Einführung in die Mikroprozessoren-Technik
von Rodnay Zaks – eine sehr gut lesbare Einführung in die faszinierende Welt der Computer, vom Microprozessor bis hin zum vollständigen System. 560 Seiten, 325 Abbildungen, Ref.Nr.: **3017** (Erscheint Herbst 1983)

VORSICHT! Computer brauchen Pflege
von Rodnay Zaks – das Buch, das Ihnen die Handhabung eines Computersystems erklärt – vor allem, was Sie damit nicht machen sollten. Allgemeingültige Regeln für die pflegliche Behandlung Ihres Systems. 240 Seiten, 96 Abbildungen, Ref.Nr.: **3013** (1983)

MEIN SINCLAIR ZX81
von D. Hergert – eine gut lesbare Einführung in diesen Einplatincomputer und dessen Programmierung in BASIC. 180 Seiten, 20 Abbildungen, Ref: **3021** (1983)

The SYBEX Library*

**Mikrocomputer-Bücher in englischer Sprache von SYBEX; lieferbar ab Lager Düsseldorf*

YOUR FIRST COMPUTER
by Rodnay Zaks 264 pp., 150 illustr., Ref. 0045
The most popular introduction to small computers and their peripherals: what they do and how to buy one.

DON'T (or How to Care for Your Computer)
by Rodnay Zaks 222 pp., 100 illustr., Ref. 0065
The correct way to handle and care for all elements of a computer system, including what to do when something doesn't work.

INTERNATIONAL MICROCOMPUTER DICTIONARY
140 pp., Ref. 0067
All the definitions and acronyms of microcomputer jargon defined in a handy pocket-size edition. Includes translations of the most popular terms into ten languages.

FROM CHIPS TO SYSTEMS:
AN INTRODUCTION TO MICROPROCESSORS
by Rodnay Zaks 558 pp., 400 illustr., Ref. 0063
A simple and comprehensive introduction to microprocessors from both a hardware and software standpoint: what they are, how they operate, how to assemble them into a complete system.

INTRODUCTION TO WORD PROCESSING
by Hal Glatzer 216 pp., 140 illustr., Ref 0076
Explains in plain language what a word processor can do, how it improves productivity, how to use a word processor and how to buy one wisely.

INTRODUCTION TO WORDSTAR™
by Arthur Naiman 208 pp., 30 illustr., Ref. 0077
Makes it easy to learn how to use WordStar, a powerful word processing program for personal computers.

DOING BUSINESS WITH VISICALC®
by Stanley R. Trost 200 pp., Ref. 0086
Presents accounting and management planning applications – from financial statements to master budgets; from pricing models to investment strategies.

EXECUTIVE PLANNING WITH BASIC
by X. T. Bui 192 pp., 19 illustr., Ref. 0083
An important collection of business management decision models in BASIC, including Inventory Management (EOQ), Critical Path Analysis and PERT, Financial Ratio Analysis, Portfolio Management, and much more.

BASIC FOR BUSINESS
by Douglas Hergert 250 pp., 15 illustr., Ref. 0080
A logically organized, no-nonsense introduction to BASIC programming for business applications. Includes many fully-explained accounting programs, and shows you how to write them.

FIFTY BASIC EXERCISES
by J. P. Lamoitier 236 pp., 90 illustr., Ref. 0056
Teaches BASIC by actual practice, using graduated exercises drawn from everyday applications. All programs written in Microsoft BASIC.

BASIC EXERCISES FOR THE APPLE
by J. P. Lamoitier 230 pp., 90 illustr., Ref. 0084
This book is an Apple version of *Fifty BASIC Exercises.*

BASIC EXERCISES FOR THE IBM PERSONAL COMPUTER
by J. P. Lamoitier 232 pp., 90 illustr., Ref. 0088
This book is an IBM version of *Fifty BASIC exercises.*

INSIDE BASIC GAMES
by Richard Mateosian 352 pp., 120 illustr., Ref. 0055
Teaches interactive BASIC programming through games. Games are written in Microsoft BASIC and can run on the TRS-80, Apple II and PET/CBM.

THE PASCAL HANDBOOK
by Jacques Tiberghien 492 pp., 270 illustr., Ref. 0053
A dictionary of the Pascal language, defining every reserved word, operator, procedure and function found in all major versions of Pascal.

INTRODUCTION TO PASCAL (Including UCSD Pascal)
by Rodnay Zaks 422 pp., 130 illustr., Ref 0066
A step-by-step introduction for anyone wanting to learn the Pascal language. Describes UCSD and Standard Pascals. No technical background is assumed.

APPLE PASCAL GAMES
by Douglas Hergert and Joseph T. Kalash 376 pp., 40 illustr., Ref. 0074
A collection of the most popular computer games in Pascal, challenging the reader not only to play but to investigate how games are implemented on the computer.

CELESTIAL BASIC: Astronomy on Your Computer
by Eric Burgess 228 pp., 65 illustr., Ref. 0087
A collection of BASIC programs that rapidly complete the chores of typical astronomical computations. It's like having a planetarium in your own home! Displays apparent movement of stars, planets and meteor showers.

PASCAL PROGRAMS FOR SCIENTISTS AND ENGINEERS
by Alan R. Miller 378 pp., 120 illustr., Ref. 0058
A comprehensive collection of frequently used algorithms for scientific and technical applications, programmed in Pascal. Includes such programs as curvefitting, integrals and statistical techniques.

BASIC PROGRAMS FOR SCIENTISTS AND ENGINEERS
by Alan R. Miller 326 pp., 120 illustr., Ref. 0073
This second book in the "Programs for Scientists and Engineers" series provides a library of problem-solving programs while developing proficiency in BASIC.

FORTRAN PROGRAMS FOR SCIENTISTS AND ENGINEERS
by Alan R. Miller 320 pp., 120 illustr., Ref. 0082
Third in the "Programs for Scientists and Engineers" series. Specific scientific and engineering application programs written in FORTRAN.

PROGRAMMING THE 6809
by Rodnay Zaks and William Labiak 520 pp., 150 illustr., Ref. 0078
This book explains how to program the 6809 in assembly language. No prior programming knowledge required.

PROGRAMMING THE 6502
by Rodnay Zaks 388 pp., 160 illustr., Ref. 0046
Assembly language programming for the 6502, from basic concepts to advanced data structures.

6502 APPLICATIONS BOOK
by Rodnay Zaks 286 pp., 200 illustr., Ref. 0015
Real-life application techniques: the input/output book for the 6502.

ADVANCED 6502 PROGRAMMING
by Rodnay Zaks 292 pp., 140 illustr., Ref. 0089
Third in the 6502 series. Teaches more advanced programming techniques, using games as a framework for learning.

PROGRAMMING THE Z80
by Rodnay Zaks 626 pp., 200 illustr., Ref. 0069
A complete course in programming the Z80 microprocessor and a thorough introduction to assembly laguage.

PROGRAMMING THE Z8000
by Richard Mateosian 300 pp., 124 illustr., Ref. 0032
How to program the Z8000 16-bit microprocessor. Includes a description of the architecture and function of the Z8000 and its family of support chips.

THE CP/M® HANDBOOK (with MP/M™)
by Rodnay Zaks 324 pp., 100 illustr., Ref. 0048
An indispensable reference and guide to CP/M – the most widely-used operating system for small computers.

MASTERING CP/M®
by Alan R. Miller 320 pp., Ref. 0068
For advanced CP/M users or systems programmers who want maximum use of the CP/M operating system ... takes up where our *CP/M Handbook* leaves off.

INTRODUCTION TO THE UCSD p-SYSTEM™
by Charles W. Grant and Jon Butah 250 pp., 10 illustr., Ref. 0061
A simple, clear introduction to the UCSD Pascal Operating System; for beginners through experienced programmers.

A MICROPROGRAMMED APL IMPLEMENTATION
by Rodnay Zaks 350 pp., Ref. 0005
An expert-level text presenting the complete conceptual analysis and desing of an APL interpreter, and actual listing of the microcode.

THE APPLE CONNECTION
by James W. Coffron 228 pp., 120 illustr., Ref. 0085
Teaches elementary interfacing and BASIC programming of the Apple for connection to external devices and household appliances.

MICROPROCESSOR INTERFACING TECHNIQUES
by Rodnay Zaks and Austin Lesea 458 pp., 400 illustr., Ref. 0029
Complete hardware and software interconnect techniques, including D to A conversion, peripherals, standard buses and troubleshooting.

Dieses Buch ist zur praktischen Anwendung bestimmt. Es soll als Werkzeug zum unmittelbaren Gebrauch am Computer verwendet werden.

sinclair

ZX Spectrum

hilft die wesentlichen Grundzüge des Programmierens beim SPECTRUM darzulegen. Programme aus dem kaufmännischen Bereich, Lehr- und Lernprogramme sowie viele Spiele helfen Ihnen, nicht nur die Grundzüge der Programmierung in BASIC mit Ihrem Spectrum zu lernen – Sie erhalten auch direkt anwendbare Programme.

Dieses Buch erweitert den Horizont der Möglichkeiten, die Ihnen mit dem SINCLAIR ZX SPECTRUM gegeben sind. Nutzen Sie Ihren Mikrocomputer optimal und haben Sie viel Freude daran.

ISBN 3-88745-022-1

SYBEX