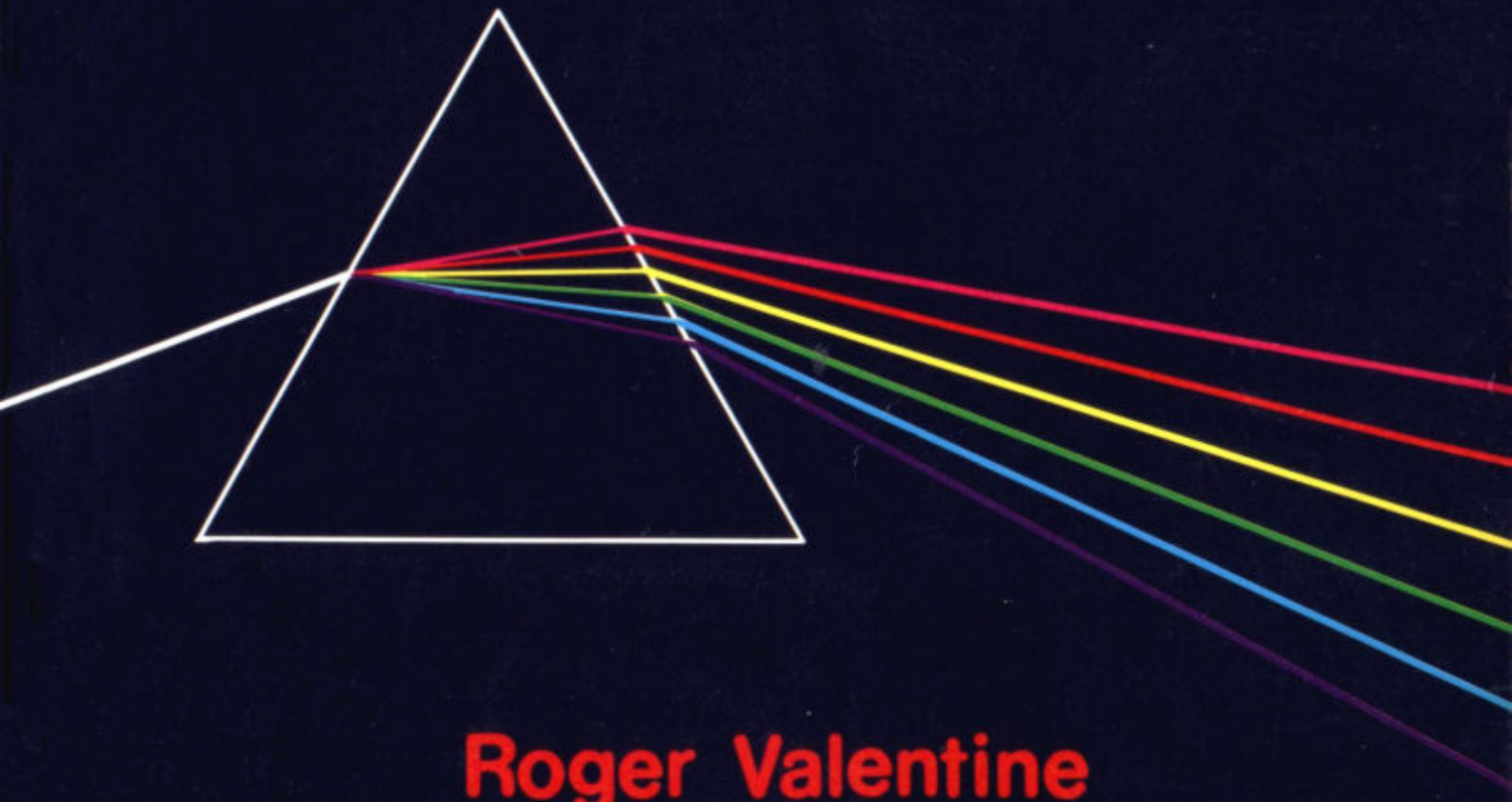
Cooperation

Hueber Software

SPECTRUM SPECTACULAR

50 Programs for the SINCLAIR SPECTRUM



Roger Valentine

ROGER VALENTINE

SPECTRUM SPEKTAKULÄR

50

Programme für den ZX SPECTRUM

Titel der englischen Originalausgabe
SPECTRUM SPECTACULAR

Übersetzung: R. Sütterlin
Umschlagfoto: Tony Stone Association, London
Umschlagentwurf: Agentur Cooperation, München
Lektorat: Joe Hembus
Herstellung: T. Lago

1. Auflage 1983 3 2 1

Satz: Fotosatz Weihrauch, Würzburg
Druck: RMO Druck, München

INHALTS-VERZEICHNIS

VORWORT

Egal, ob Sie als Neuling an den ZX Spectrum herangehen oder als versierter Programmierer, fasziniert von den fantastischen Grafik-, Farb- und Speichermöglichkeiten des Spectrum, sofort in die Feinheiten einsteigen – hier finden Sie eine Reihe von Programmen und Routinen, die Ihnen nützlich sein sollten, wenn Sie alle Qualitäten Ihres Computers voll ausschöpfen wollen.

Noch ein Hinweis, um Ihnen unnötige Fehler bei der Eingabe der Programme zu ersparen: Der besseren Lesbarkeit zuliebe wurden die Listings auf einem Normalpapier-Drucker ausgedruckt. Dieser Printer stellt einige Zeichen etwas anders dar, als Sie es vom Bildschirm oder dem kleinen ZX-Printer gewohnt sind. Bitte sehen Sie sich die folgenden Zeichen und ihren Ausdruck an.

```
1    = Zahl eins
I    = grosser Buchstabe I
l    = kleiner Buchstabe L
0    = Zahl null
O    = grosser Buchstabe O
o    = kleiner Buchstabe O
q    = kleiner Buchstabe Q
a    = kleiner Buchstabe A
()   = Klammern
<>   = kleiner als; groesser als
""   = der 'Null String' (2 An-
fuehrungszeichen)
" "= Anfuehrungszeichen; space;
Anfuehrungszeichen
```

Bei den seltenen Gelegenheiten, wo eine Kette von zwei oder mehr Leerzeichen benötigt wird, wurde eine REM-Zeile eingefügt, um die Anzahl der Leerräume anzugeben.

Alle Grafikzeichen (einschließlich der selbst definierten) wurden mit Hilfe der entsprechenden CHR$-Befehle erzeugt. Das erspart Ihnen sinnlose Probeläufe durch schlecht lesbare Listings.

Alle Variablennamen wurden klein geschrieben. Alle

Jedes Programm enthält eine Zeile 9000, die gewöhnlich so aussieht:
9000 SAVE "Programmname" LINE 0

Wenn Sie ein Programm eingegeben und sich davon überzeugt haben, daß das Programm fehlerfrei ist, können Sie GO TO 9000 eingeben. Dann wird das Programm so auf Band gespeichert, daß es sich nach dem Zurückladen von selbst startet.

VORBEMERKUNG ZU DEN LISTINGS

Übrigens, alle Programme laufen auf der 16K Version des Spectrum, außer: "Rechnung", "Buchen" und dem Maschinenprogramm "Blättern", die 48K benötigen.

5 LEICHTE STÜCKE

GROSSE BUCHSTABEN
CHR-TEST
FARBTAFEL
KLAVIER
DIGITAL UHR

GROSSE BUCH-STABEN

Die ersten beiden Programme untersuchen den Zeichensatz des ZX Spectrum.

Der gesamte Zeichensatz, der im Anhang A des Spectrum-Handbuchs aufgezeichnet ist, läßt sich in fünf Gruppen einteilen.

CHR$ 0 bis CHR$ 31 : Kontrollzeichen

Das sind keine 'Zeichen' im üblichen Sinn, da sie nicht auf dem Schirm ausgegeben werden. Sie verursachen solche Dinge wie Farbe und Position des Zeichens, das als nächstes angegeben wird.

CHR$ 32 bis CHR$ 127: "Normale Zeichen"

Mit nur zwei kleinen Ausnahmen, (dem Pfund und dem COPY Zeichen) ist dies der berühmte ASCII Zeichensatz, der bei den meisten Personalcomputern verwendet wird.

CHR$ 128 bis CHR$ 143 : Sinclair Graphikzeichen
CHR$ 144 bis CHR$ 164 : vom Benutzer zu definierende Graphik
CHR$ 165 bis CHR$ 255 : "Zusammengesetzte" Zeichen

Schlüsselwörter usw., die alle eine spezielle Funktion haben, aber die auf dem Schirm durch eine Kombination von zwei oder mehr Zeichen dargestellt werden.

Das erste Programm betrachtet lediglich den "normalen"

abgespeichert sind, indem es sie 64mal vergrößert am Bildschirm anzeigt.

Die Zeichen sind im ROM unter einer Adresse abgespeichert, auf die die Systemvariable CHARS zeigt (siehe Zeile 10), und zwar auf die gleiche Weise, wie durch den Benutzer definierte Graphikzeichen. Acht Bytes bestimmen die 'Bildpunkt-Matrix' jedes Zeichens.

Diese 8 Bytes werden in Zeile 25 identifiziert und in Zeile 40 in binäre Form umgewandelt (BIT). Jedes BIT, das normalerweise dazu benutzt werden würde, um anzuzeigen, daß ein einzelner Bildpunkt 'geplottet' bzw. nicht 'geplottet' wird, wird nun benutzt, um ein ganzes Zeichen zu erzeugen. Entweder ein schwarzes Quadrat (CHR$ 143), wenn der Wert des BITs 1 ist oder ein weißes Quadrat (CHR$ 144), wenn sein Wert Ø ist.

Der Zweck der durch den Benutzer definierten Graphikzeichen in diesem Programm besteht darin, einen Raster zu erzeugen, damit man genau erkennen kann, wie jedes einzelne Zeichen aufgebaut ist. Es kann für Sie beim Entwerfen Ihrer eigenen Zeichen nützlich sein, genau zu wissen, wie Sinclair die Buchstaben entwickelt hat.

```
    5 RESTORE : FOR j=0 TO 7: REA
D a: POKE USR CHR$ 144+j,a: NEXT
 j
   10 LET c=PEEK 23606+256*PEEK 2
3607+256
   15 FOR j=0 TO 95: CLS : PRINT
AT 4,20;"CHR$ ";j+32;" ";CHR$ (j
+32)
   20 FOR p=0 TO 7
   25 LET byte=PEEK (c+p+j*8)
   30 PRINT AT p,15;byte
   35 FOR l=1 TO 8
   40 LET x=INT (byte/2): LET bit
=byte-2*x: LET byte=x
   45 PRINT AT p,9-l;CHR$ (144-bi
t)
   50 NEXT l
   55 NEXT p
   60 IF j<>95 THEN PRINT AT 15,0
; FLASH 1;"Bitte beliebige Taste
 druecken": PAUSE 0
   65 NEXT j: STOP
  100 DATA 255,129,129,129,129,12
```

CHR TEST

Welchen einfacheren Weg, um Sie an den Zeichensatz des Spectrum zu gewöhnen, könnte es geben, als ein einfaches Spiel, das Ihr Wissen über den CODE jedes einzelnen Zeichens testet.

Das Programm zeichnet eine stylisierte Bombe mit schnell abbrennender Lunte. Es zeigt außerdem ein zufällig ausgewähltes Zeichen, und es bleiben Ihnen einige wenige Sekunden, um die richtige CODE Nummer einzugeben und so die Bombe zu 'entschärfen'.

Am Anfang werden Sie vermutlich völlig versagen, sogar mit Hilfe des Spectrum-Handbuches, aber nach einer Weile sollten Sie das Spiel leichter finden, weil Sie allmählich den ASCII-Teil des Zeichensatzes lernen und vielleicht sogar den Graphik- und den zusammengesetzten Teil. (Aus einleuchtenden Gründen unterscheidet das Programm **nicht** zwischen Kontrollzeichen, benutzerdefinierten Zeichen und 'normalen' ASCII-Zeichen.)

Das Interessanteste an dem Programm ist die Art, wie Eingaben gemacht werden. Beachten Sie, daß das entsprechende Befehlswort INPUT überhaupt nicht vorkommt. Die hier benutzte Technik wird ausführlich im 'Unterprogramm-Kapitel' ("Routinen") beschrieben. Der Grund, warum diese Technik verwenden wurde: INPUT veranlaßt den Rechner zu warten, bis der Benutzer ENTER drückt, bevor er im Programm fortfährt. Das ist natürlich in einem Fall wie diesem, wo ein Zeitlimit für die Eingabe besteht, völlig unpassend.

Der Befehl OVER 1 wird in dem Programm dazu verwendet, um die Lunte Bildpunkt für Bildpunkt zu verkürzen (Zeile 500) und um falsche Eingaben zu löschen (Zeile 105).

Übrigens: Wenn Sie zunächst eine längere Zündschnur benötigen, dann erhöhen Sie einfach den Wert von t (Zeile 25) auf 15 oder 20.

```
  10 BORDER 6: PAPER 7: INK 2: C
LS
  15 PRINT AT 15,5;"BOMBE"
  20 CIRCLE 56,52,39
  25 LET T=10
  30 PLOT 56,91: DRAW 0,5: DRAW
T,T: PLOT OVER 1;56+T,96+T
  35 LET X=INT (RND*223)+33
  40 IF X>=144 AND X<=164 THEN G
O TO 35
  45 PRINT AT 0,0;"Was ist der C
ode fuer:";CHR$ x
  50 LET a$="": LET a=0
  55 PAUSE 10: LET i$=INKEY$
  60 GO SUB 500
  65 IF t=0 THEN GO TO 950
  70 IF i$=CHR$ 13 THEN GO TO 10
0
  75 IF i$<"0" OR i$>"9" THEN GO
 TO 55
  80 LET a$=a$+i$: LET a=VAL a$:
 PRINT AT 2,22;a$: GO TO 55
 100 IF a=x THEN GO TO 900
 105 PRINT AT 2,22; OVER 1;a$: G
O TO 50
 500 DRAW OVER 1;-1,-1
 510 LET t=t-1: RETURN
 900 PRINT AT 5,11; FLASH 1;"GUT
 GEMACHT!!": GO TO 1000
 950 FOR k=1 TO 2: FOR j=0 TO 7:
 PAPER j: BORDER j: CLS : BEEP .
01,k*7-j: NEXT j: NEXT k: BORDER
 6
1000 PRINT "Der Code fuer ";CHR$
 x;" ist: ";x
1010 PRINT AT 21,0;"""R"" zum We
itermachen druecken"
1020 IF INKEY$="r" OR INKEY$="R"
 THEN RUN
1030 GO TO 1020
9000 SAVE "CHR TEST" LINE 0
```

FARBTAFEL

Bei Ihren ersten Versuchen mit dem Spectrum haben Sie vielleicht gemerkt, daß einige Kombinationen der INK und PAPER Farben einfach nicht funktionieren. Alles, was beispielsweise in himmelblau (cyan) auf scharlachrot (magenta) geschrieben wird, erscheint uns als unleserliches Gekleckse!

Dieses Programm zeigt Ihnen genau, welche Kombinationen funktionieren (überraschend wenige), und es dient außerdem noch einem weit wichtigeren Zweck. Es führt die Attributnummer, die zu jeder einzelnen Kombination gehört, in tabellarischer Form auf. (d.h. „die oben erwähnte himmelblaue INK/ scharlachrote PAPER Kombination hat die Attributnummer (ATTR) 29).

Sie werden diese Tabelle brauchen, wenn Sie Programme schreiben, in denen die ATTR-Funktion verwendet wird, oder wenn Sie Werte in die Attribut-Datei einPOKEn.

Beachten Sie, daß, wenn Sie das 'global' Kommando BRIGHT 1 (Zeile 10) verwenden und dann alles mit BRIGHT 0 PRINTen, (Zeilen 30, 40 u. 70), weiße Zeichen und sogar weißes PAPER sich deutlich von einem weißen Hintergrund abheben.

Um helle (BRIGHT) Zeichen zu erzeugen, addieren Sie 64 zu dem angezeigten ATTR-Wert, um sie blinken zu lassen, addieren Sie 128.

```
   10 BRIGHT 1: CLS : PRINT TAB 1
5;" PAPER "';"';"';"';"';"';"I";"';"N";"';"K
"
   20 FOR j=0 TO 7
   30 PRINT AT 2,j*3+5; PAPER j;
BRIGHT 0;CHR$ 32;CHR$ 32
   40 PRINT AT j*2+5,2; PAPER j;
BRIGHT 0;CHR$ 32
   50 FOR k=0 TO 7
   60 LET a$=STR$ (8*j+k): IF LEN
```

```
T  0;AT  k*2+5,j*3+6;a$
   80  NEXT  k
   90  NEXT  j
  100  STOP
 9000  SAVE  "ATTR TAFEL" LINE 0
```

KLAVIER

Es ist eine Kleinigkeit, die Tastatur des Spectrum so umzudefinieren, daß die Tasten andere Zeichen, als die darauf abgebildeten erzeugen. Sie können sogar erreichen, daß sie überhaupt keine Zeichen, sondern Musiknoten produzieren.

Dieses Programm verändert die obere Buchstabenreihe der Tastatur (QWERTYUIOP) so, daß die 'weißen Töne' eines Klaviers (vom mittleren C aufwärts, wobei die Taste "E" den Ton E erzeugt) produziert werden, und die entsprechend plazierten Zahlentasten die 'schwarzen Töne' erzeugen. (Wenn Sie eine richtige Klaviertastatur betrachten, werden Sie feststellen, daß die Spectrumtaste "3" D+ entspricht, aber daß "4" kein entsprechendes Gegenstück hat und deshalb undefiniert geblieben ist).
Das Programm erzeugt einen neuen 'Zeichensatz' im Array a( ). Es handelt sich dabei um einen äußerst knappen Zeichensatz mit keinem Zeichen für die meisten Tasten und den Tonwerten 1–17 für die in Zeile 10 angegebenen Tasten. (Die Berichtigungen von +1 in den Zeilen 30 und 50 und –1 in Zeile 60 sind nötig, weil Tabellenelemente bei 1 und Zeichensätze bei 0 beginnen).

Mit Zeile 60 werden Eingaben von Tasten, für die kein entsprechender Tonwert existiert, zurückgewiesen, und ein Ton von ½ Sekunde Dauer für die 'Klaviertasten' erzeugt.

```
   10 LET n$="q2w3er5t6y7ui9o0p"
   20 DIM a(256)
   30 FOR j=1 TO 17: LET a(CODE n
$(j)+1)=j
   40 NEXT j
   50 LET i=CODE INKEY$+1
   60 IF a(i) THEN BEEP .5,a(i)-1
   70 GO TO 50
9000 SAVE "KLAVIER" LINE 0
```

UHR

Das Spectrum-Handbuch erklärt ein Uhrenprogramm, mit dem Zeichnen des Zifferblattes, der Zeiger usw. Eigenartig, daß Sinclair, einer der Pioniere der Digitaluhr, uns nun dazu ermuntert, seinen Computer als Uhr mit Analoganzeige zu verwenden!

Um wieviel angemessener (und einfacher) ist es, eine Digitaluhr zu bauen.

Wir verzichten darauf, das Programm in seinen Einzelheiten zu erklären, da die angewandten Prinzipien genau die gleichen sind, wie im Uhrenprogramm des Handbuchs (Kapitel 18) erklärt.

Zeilen 10 bis 70 wandeln die eingegebenen Stunden und Minuten in 3 Werte um, die in die Systemvariable FRAMES hineingePOKEt werden, und die Zeilen 100 bis 140 führen die umgekehrte Operation durch, nämlich die Umwandlung der Werte in FRAMES, in Stunden, Minuten und Sekunden.

Der Rest des Programms legt bloß noch das Ausgabeformat fest (Zeile 180).

Das Programm erzeugt eine Uhr mit 24-Stunden-Anzeige. Wenn Sie so altmodisch sind, daß Sie eine 12-Stunden-Anzeige vorziehen, dann ändern Sie den Befehl 2 in Zeile 140 so ab, daß er lautet:

IF h=13 THEN LET h=1

WICHTIG:

Wenn Sie in einem Land leben, wo die Wechselstromfrequenz 60 Hertz beträgt, dann ersetzen Sie "50" durch "60" in Zeile 30 und 100.

```
   1 BORDER 6: PAPER 6: INK 0: C
LS
   5 DIM t$(8): LET t$(3)=":": L
ET t$(6)=":"
  10 INPUT "Stunde ? ";h
  20 INPUT "Minute ? ";m
  30 LET a=(h*3600+m*60)*50
  40 LET b=INT (a/65536): POKE 2
3674,b
  50 LET c=a-b*65536
  60 LET a=INT (c/256): POKE 236
73,a
  70 LET b=c-a*256: POKE 23672,b
 100 LET t=INT ((65536*PEEK 2367
4+256*PEEK 23673+PEEK 23672)/50)
 110 LET h=INT (t/3600)
 120 LET b=t-h*3600
 130 LET m=INT (b/60)
 140 LET s=b-m*60: IF h=24 THEN
LET h=0
 150 LET t$( TO 2)=STR$ h: IF LE
N STR$ h=1 THEN LET t$( TO 2)="0
"+STR$ h
 160 LET t$(4 TO 5)=STR$ m: IF L
EN STR$ m=1 THEN LET t$(4 TO 5)=
"0"+STR$ m
 170 LET t$(7 TO )=STR$ s: IF LE
N STR$ s=1 THEN LET t$(7 TO )="0
"+STR$ s
 180 PRINT AT 11,12;t$
 190 GO TO 100
9000 SAVE "UHR" LINE 0
```

FRAGE: Was ist bei diesem Listing verkehrt?

```
   5 CLEAR 29999
  10 LET c=PEEK 23606+256*PEEK 2
3607+256
  20 FOR i=0 TO 95
  30 FOR b=0 TO 7
  40 POKE 30000+7-b+8*i,PEEK (c+
b+8*i)
  50 NEXT b
  60 NEXT i
  70 POKE 23606,48: POKE 23607,1
16
  80 STOP
9000 SAVE "INVERT" LINE 0
9995 REM Zurueckschalten
9996 POKE 23606,0: POKE 23607,60
```

ANTWORT: Überhaupt nichts! – Wenn Sie wissen wollen, wie es gemacht wird, dann blättern Sie bitte weiter

SPIEL ES NOCH EINMAL, SAM

ÜBUNGS-SPIEL
SQUASH
FESTUNG/FORT-DATA
MINENFELD

ÜBUNGS-SPIEL

Wenn Sie ab und zu in einen Spielsalon gehen und sich die Typen ansehen, die vor den Videoschirmen sitzen, werden Sie feststellen, daß manche von ihnen geradezu unglaublich hohe Punktzahlen erreichen. Sicher gehört dazu eine spezielle Begabung, aber glauben Sie mir: Auch Sie können zumindest Ergebnisse erreichen, die Ihnen eine Blamage ersparen.

Dieses Spiel simuliert die Grundzüge aller Schläger-und-Ball-Videospiele. Es gibt keine Punkte und kein Spielende, Sie müssen nur versuchen, die Bälle zu treffen, die Ihnen laufend zugeworfen werden.

Das ist jedoch nicht so leicht wie es klingt. Durch das Fehlen der Wertung usw. läuft das Spiel schneller ab als die meisten der in BASIC geschriebenen Spiele, (obwohl ich die Toneffekte beibehalten habe; wenn das Spiel noch schneller laufen soll, können Sie diese ebenfalls herausnehmen). Ebenfalls sind die Bedingungen für das 'Treffen' des Balles ziemlich schwer. Sie müssen den Ball mit der vollen Fläche des Schlägers treffen, Kantentreffer sind nicht erlaubt. (Diese Regel wird im nächsten Programm "SQUASH" gelockert, wenn Sie also erst das Übungsspiel beherrschen, sollten Sie alle Gegner im Squash besiegen können.)

Die Struktur des Programms ist ausreichend, daß Sie daraus jedes der Videospiele der ersten Generation entwickeln können (BREAKOUT usw.).

Die Ball-Koordinaten (x, y) werden durch die zwei Variablen ud (up/down) und lr (links/rechts) verändert. Jede davon kann entweder den Wert -1 (hoch, links) oder +1

wand ändert das Vorzeichen von lr, die Berührung der Decke oder des Schlägers das von ud.

Die Schlägerkoordinaten (a, b), von denen a eine Konstante ist, werden durch die INKEY$-Routine in Zeile 100 verändert.

In beiden Fällen benutze ich die Technik, einen Leerraum an der Stelle zu produzieren, an der vorher der Schläger bzw. der Ball war. Wenn Sie wollen, können Sie natürlich auch OVER 1 verwenden, um den alten Ball oder Schläger zu entfernen.

Damit der Schläger eine höhere Geschwindigkeit als der Ball erreichen kann, brauchen Sie nur das 'Schläger-Bewegungs-Unterprogramm' ZWEIMAL während jeder Bewegung des Balles aufzurufen. (Zeilen 30 und 45).

```
    5 BORDER 4: PAPER 4: BRIGHT 1
: INK 9
   10 CLS
   15 LET a=20: LET b=15: LET w=0
   20 LET x=0: LET y=INT (RND*31)
: LET ud=1: LET lr=INT (RND*2):
IF lr=0 THEN LET lr=-1
   25 PRINT AT x,y;".": IF w THEN
 BEEP .01,30: LET w=0
   30 GO SUB 100
   35 LET nx=x+ud
   40 IF nx=0 THEN LET ud=-ud: LE
T w=1
   45 GO SUB 100
   50 LET ny=y+lr: IF ny=0 OR ny=
31 THEN LET lr=-lr: LET w=0
   55 IF nx=a AND ny=b THEN LET u
d=-ud: BEEP .01,50
   60 PRINT AT x,y;CHR$ 32: LET x
=nx: LET y=ny
   65 IF x=22 THEN BEEP .3,0: GO
TO 20
   70 GO TO 25
  100 LET n=b+(INKEY$="8" AND b<3
1)-(INKEY$="5" AND b>0)
  105 IF n<>b THEN PRINT AT a,b;C
HR$ 32: LET b=n
  110 PRINT AT a,b;CHR$ 95
  115 RETURN
9000 SAVE "UEBUNG" LINE 0
```

SQUASH

Obwohl es eindeutig auf dem vorhergehenden Programm aufbaut, ist dies ein ausgereiftes Spiel für 2 Spieler. (Wenn Sie unsozial eingestellt sind, könnten Sie es vorziehen mit Ihrer linken Hand gegen Ihre Rechte zu spielen – sehr schwierig).

Offensichtlich liegen die Hauptunterschiede zwischen den beiden Spielen in der Schlägerbewegungsroutine. Um zwei Schläger unabhängig voneinander bewegen zu können, ist die INKEY$-Methode von Haus aus ungeeignet, da sie nicht in der Lage ist, eine gedrückte Taste zu identifizieren, wenn gleichzeitig mehrere Tasten gedrückt werden. Die IN-Funktion jedoch, obwohl etwas schwieriger zu benutzen, hat den gleichen Erfolg und kann außerdem alle Tasten – auch wenn gleichzeitig gedrückt – identifizieren.

Es gibt acht separate IN-Befehle, um die gesamte Tastatur zu lesen. Dieses Programm benutzt nur zwei davon:

IN 63486 um die Tasten 1 bis 5 zu lesen
IN 61438 um die Tasten 6 bis Ø zu lesen

Die Tasten 1 und 2 werden benutzt, um den Schläger von Spieler 1 nach links bzw. rechts zu bewegen, und 9 und Ø zum Bewegen des Schlägers von Spieler 2. In den Zeilen 105 und 110 werden die beiden IN-Befehle genau wie ein INKEY$ Befehl benutzt (außer daß IN eine Zahl und INKEY$ eine Zeichenkette ausgibt). Die beiden IN-Befehle überlagern sich nicht.

IN 61438 gibt einen Wert von 253, wenn "9" gedrückt wird, und von 254, wenn "Ø" gedrückt ist, unabhängig davon, ob irgendwelche anderen Tasten auf irgend einem Bereich der Tastatur gedrückt werden. Ebenso gibt IN 63486 einen Wert von 253 für Taste "2" und von 254 für Taste "1" aus.

(In Wirklichkeit hängen die absoluten Werte, die durch IN erzielt werden, außer von gedrückten Tasten noch von anderen Faktoren ab. Vor allem vom Status des EAR-Steckers, welcher durch den BEEP-Befehl beeinflußt wird. Mit Zeile 35 wird daher sichergestellt, daß die ausgegebenen Werte in der richtigen Größenordnung sind.)

Veränderungen im 'Ballbewegungs-Teil' sind weniger bedeutsam und mußten gemacht werden, um das Spiel interessanter zu gestalten, als eine bloße 2-Spieler-Version des Übungsspiels.

'Kantenschläge' sind jetzt erlaubt (Zeile 70) und verändern das Vorzeichen der lr-Variablen. Das macht das Spiel einfacher, da der Schläger dadurch um das Dreifache größer wird, und gibt andererseits Gelegenheit, die Bälle möglichst ungünstig für den Gegner zu plazieren.

Um das zu erschweren, wird das Unterprogramm für die Schlägerbewegung nur einmal pro Ballbewegung aufgerufen. Sie sind in diesem Spiel nicht schneller als der Ball, aber denken Sie daran, Sie können Ihren Schläger sogar bewegen, wenn Ihr Gegner am Zug ist. Natürlich können Sie den Ball NICHT treffen, wenn Ihr Gegner schlägt, weil die Schlägerkoordinaten in einem zweistufigen Array abgelegt sind und die Zeilen 75 und 80 nur im entsprechenden Feld nachsehen, wenn sie einen Treffer untersuchen.

Die Ergebnisse werden auf dem Schirm angezeigt, und ein Ergebnis von 21 bei einem Spieler läßt diesen das Spiel gewinnen.

```
    5 BORDER 7: PAPER 6: INK 9: C
LS
   10 DIM s(2): DIM b(2): DIM n(2
): LET w=0: LET b(1)=13: LET b(2
)=17: LET s(1)=-1: LET p=0
   15 LET x=0: LET y=INT (RND*31)
: LET ud=1: LET p=p+1: IF p=3 TH
EN LET p=1
   20 LET lr=INT (RND*3)-1: IF NO
T lr THEN GO TO 20
   25 LET s(p)=s(p)+1
   30 PRINT AT 21,0;"Am Spiel: ";p
```

```
 GO TO 1000
  35 OUT 63486,255: OUT 61438,25
5
  40 PRINT AT x,y;".": IF w THEN
 BEEP .01,30: LET w=0
  45 GO SUB 100
  50 LET nx=x+ud
  55 IF nx=0 THEN LET ud=-ud: LE
T w=1
  60 LET ny=y+lr: IF ny=0 OR ny=
31 THEN LET lr=-lr: LET w=1
  65 IF ud=-1 OR nx<18 THEN GO T
O 85
  70 IF nx=17+p AND ABS (ny-b(p)
)=1 THEN LET ud=-ud: LET lr=ny-b
(p): BEEP .01,50: LET p=p+1: IF
p=3 THEN LET p=1
  75 IF nx=17+p AND ny=b(p) THEN
 LET ud=-ud: BEEP .01,50: LET p=
p+1: IF p=3 THEN LET p=1
  80 PRINT AT 21,9;p
  85 PRINT AT x,y;CHR$ 32: LET x
=nx: LET y=ny
  90 IF x=21 THEN BEEP .3,0: GO
TO 15
  95 GO TO 40
 100 PRINT AT 18,b(1);CHR$ 95;AT
 19,b(2);CHR$ 95
 105 LET n(2)=b(2)+(IN 61438=254
 AND b(2)<31)-(IN 61438=253 AND
b(2)>0)
 110 LET n(1)=b(1)+(IN 63486=253
 AND b(1)<31)-(IN 63486=254 AND
b(1)>0)
 115 IF n(1)=b(1) THEN GO TO 125
 120 PRINT AT 18,n(1);CHR$ 95;AT
 18,b(1);CHR$ 32: LET b(1)=n(1)
 125 IF n(2)=b(2) THEN RETURN
 130 PRINT AT 19,n(2);CHR$ 95;AT
 19,b(2);CHR$ 32: LET b(2)=n(2):
 RETURN
1000 PRINT AT 5,0; PAPER 7; FLAS
H 1;"SPIELENDE :Druecke R zum we
iter-spielen"
1005 IF INKEY$="r" THEN RUN
1010 GO TO 1005
9000 SAVE "SQUASH" LINE 0
```

FESTUNG/ FORT-DATA

"Festung" ist mehr ein intellektuelles Puzzle als ein Spiel im üblichen Sinn. Es ist sehr einfach, wenn man das Geheimnis kennt, und praktisch unmöglich zu gewinnen, wenn man es nicht kennt. Der Spectrum jedenfalls kennt es und spielt daher sehr gut!

Die beiden Programme sollten wie folgt eingegeben werden:

Geben Sie "FORT-DATA" ein.
RUN. Dadurch werden die Arrays aufgebaut, aber außer der STOP-Meldung erscheint nichts auf dem Schirm.
Geben Sie CONT ein, um eine Kopie von FORT-DATA zusammen mit seinen Arrays auf Band zu speichern.
Geben Sie **NICHT** NEW ein!
Tippen Sie einfach "FESTUNG" ein und überschreiben damit die Zeilen von "FORT-DATA".
Geben Sie **NICHT** RUN ein! Benutzen Sie GO TO 9000, um eine Kopie von FESTUNG zusammen mit den Arrays von FORT-DATA abzuspeichern. Diese Kopie läuft automatisch ab, wenn sie ohne CLEAR geladen wird. Sie können das Programm auch sofort starten, jedoch **NICHT MIT** RUN sondern durch GO TO 0.

Haben Sie irrtümlich RUN (oder CLEAR) beim Eintippen von "FESTUNG" eingegeben, so ist das auch kein Beinbruch. Geben Sie einfach das Programm weiter ein, dann:
Geben Sie GO TO 9000 zum Speichern ein
Geben Sie LOAD "FORT-DATA" ein
Laden Sie "FORT-DATA" vom Band, und es läuft ab und zeigt die STOP-Meldung wie vorher.
Jetzt geben Sie MERGE "FESTUNG" ein.
Schließlich geben Sie GO TO 9000 ein, um eine weitere Kopie von "FESTUNG" abzuspeichern – dieses Mal mit

Warum das alles? Warum werden nicht beide Programme zu einem zusammengefaßt?
Der Grund besteht NICHT darin, daß Speicherplatz gespart werden soll, obwohl das durch diese Technik geschieht. Sie sollten sich auch daran erinnern, wenn Sie einmal längere Programme schreiben. Vielmehr sind manche Leute heutzutage so 'computer-gelehrt', daß traditionelle Problemlösungsmethoden einfach abgelehnt werden. Mit einem 'Quiz-Programm' konfrontiert würden sich viele Leute nicht die Mühe machen, die Fragen auf die übliche Art zu beantworten, sondern einfach das Programm BREAKen und im Listing nach den gesuchten Antworten schauen. Ich habe viele kommerzielle Quiz-Programme gesehen, wo Fragen und Antworten tatsächlich in der gleichen Zeile des Programms stehen!

Die Moral davon ist, daß es nicht mehr länger genügt, die Programme 'idiotensicher' zu machen, sie müssen außerdem auch noch 'geniesicher' sein!

In "FESTUNG" ist die Lösung nicht aus dem Listing ersichtlich, sondern der Schlüssel ist in den Zahlen enthalten, die im Array c() abgelegt sind. Bei Anwendung dieser Zwei-Programm-Technik ist es möglich, auf dieses Array zuzugreifen, ohne daß sein Inhalt im Listing offenbart wird.

SPIELREGELN

FESTUNG kann von zwei Spielern oder einem Spieler und dem Computer gespielt werden. Die beiden Spieler sind der ANGREIFER und der VERTEIDIGER. Das Spielfeld stellt ein Schloß dar mit 18 Räumen, die mit den Buchstaben A bis R markiert sind. Jeder Raum hat 4 Ausgänge, von denen jeder zu einem angrenzenden Zimmer führt. (Ein Plan des Schlosses wird am Bildschirm gezeigt, um es näher zu erklären). Das Ziel eines jeden Spielers ist es, den Gegner durch Betreten des Raumes, in dem sich dieser befindet, gefangenzunehmen.

Es gibt gewisse Beschränkungen bei den Bewegungen der

Der Verteidiger darf den 'Hauptturm', d.h. das Dreieck, das durch die Räume A, B und D gebildet wird. nicht verlassen.

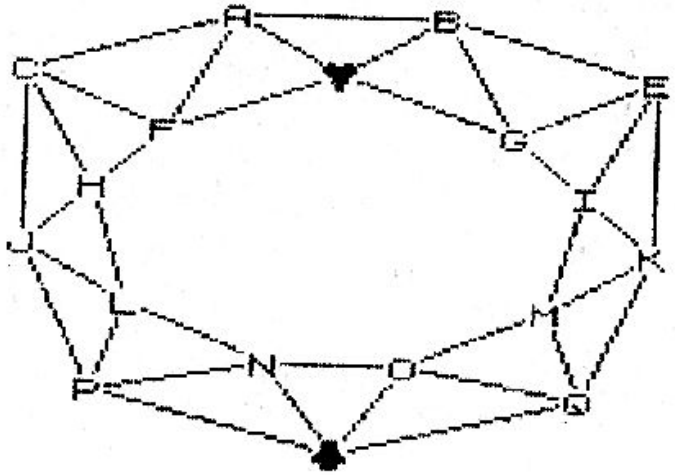
Der Angreifer darf jeden Raum betreten, aber er kann jeden Durchgang nur einmal benutzen (die Wege werden vom Schirm gelöscht, sobald sie einmal begangen wurden.) Es ist daher möglich, daß der Angreifer das Spiel verspielt, weil er in einem Raum gefangen ist, dessen vier Ausgänge bereits benutzt worden waren. Der Angreifer muß IMMER zuerst ziehen.

So betrachtet, hat der Verteidiger eine ziemlich langweilige Rolle. Er kann nur in den drei Räumen des Hauptturms umhergehen und darauf warten, daß der Angreifer in seine Reichweite gelangt. Versuchen Sie, ein paar Spiele gegen einen Freund zu spielen, (lassen Sie ihn den Verteidiger spielen!), und Sie werden ihn vermutlich leicht schlagen. Recht bald wird er die Rolle des Angreifers übernehmen wollen. Lassen Sie ihn, aber lassen Sie den Computer verteidigen. Ich kann **GARANTIEREN**, daß der Computer gewinnen wird!

Die Verteidigung des Computers bei "FESTUNG" ist unschlagbar, und sein Angriff, obwohl abzuwehren, ist sehr gut. Um gewinnen zu können, müssen Sie den Verteidiger spielen, und Sie müssen entweder eine Menge Glück haben oder die gleiche Strategie übernehmen, die der Computer bei der Verteidigung anwendet. Ich werde Ihnen NICHT erzählen, worin diese Strategie besteht, aber ich habe Ihnen bereits gesagt, wo der Schlüssel zu finden ist, und der Rest hängt von Ihnen ab.

```
   10 DIM p(18,2): DIM q(18,2)
   20 RESTORE : FOR j=1 TO 18: RE
AD p(j,1),p(j,2)
   30 LET q(j,1)=4+p(j,2)*6: LET
q(j,2)=172-p(j,1)*8
   40 NEXT j
   50 DIM c(18): FOR j=1 TO 18: R
EAD c(j): NEXT j
   60 DIM l$(18,4): FOR j=1 TO 18
```

```
  70 STOP
1000 DATA 1,12,1,18,3,6,3,15,3,2
4,5,10,5,20,7,8,7,22,9,6,9,24,11
,9,11,21,13,13,13,17,14,8,14,22,
16,15
2000 DATA 1,2,3,3,3,2,1,1,2,2,1,
3,3,2,1,1,2,3
3000 DATA "BCDF","ADEG","AFHJ","
ABFG","BGIK","ADCH","BDEI","CFJL
","GEKM","CHLP","EIMQ","HJPN","I
KQO","LOPR","MNQR","JLNR","MKOR"
,"NOPQ"
9000 SAVE "FORT-DATA" LINE 0
```




```
   1 GO TO 100
   5 CLS : FOR j=1 TO 18
  10 FOR k=1 TO 4
  15 PLOT INK 4;q(j,1),q(j,2)
  20 IF l$(j,k)>CHR$ (j+64) THEN
 GO TO 35
  25 DRAW INK 4;q(CODE l$(j,k)-6
4,1)-q(j,1),q(CODE l$(j,k)-64,2)
-q(j,2)
  30 NEXT k
  35 NEXT j
  40 FOR j=1 TO 18: PRINT AT p(j
,1),p(j,2);CHR$ (j+64): NEXT j
  50 RETURN
  60 PRINT AT 21,0;e$: FOR j=1 T
O 100: NEXT j: PRINT AT 21,0; OV
ER 1;e$: RETURN
  70 PRINT INK 2;AT p(ap,1),p(ap
,2);CHR$ 144: RETURN
  75 PRINT INK 1;AT p(dp,1),p(dp
,2);CHR$ 145: RETURN
 100 RESTORE : FOR j=USR CHR$ 14
4 TO USR CHR$ 145+7: READ a: POK
E j,a: NEXT j
 150 BORDER 6: PAPER 6: BRIGHT 1
: INK 9: CLS
 160 DIM m$(18,4): FOR j=1 TO 18
: LET m$(j)=l$(j): NEXT j
 170 LET ap=18: LET dp=4
 180 DIM e$(20)
 190 LET e$="": LET q=0: RANDOMI
```

```
";AT 5,0;"1: 1 Spieler"''"2: 2 S
pieler"
  210 LET i$=INKEY$: IF i$<"1" OR
  i$>"2" THEN GO TO 210
  220 LET no=0: IF i$="1" THEN GO
  SUB 3000
  230 GO SUB 5: GO SUB 70: GO SUB
  550
  500 IF no<>100 THEN GO TO 1000
  510 GO TO 8500
  520 IF no<>97 THEN GO TO 2000
  530 GO TO 8000
1000 PRINT AT 21,0;"Angreifer am
 Zug (";  INK 2;CHR$ 144;  INK 9;")
"
1010 IF m$(ap)="****" THEN LET w
$="VERTEIDIGER": GO TO 7000
1020 LET i$=INKEY$
1030 IF i$<"a" OR i$>"r" THEN GO
 TO 1020
1040 LET e$="KEIN DURCHGANG"
1050 FOR j=1 TO 4
1060 IF m$(ap,j)=CHR$ (CODE i$-3
2) THEN GO TO 1100
1070 IF m$(ap,j)=CHR$ 42 AND l$(
ap,j)=CHR$ (CODE i$-32) THEN LET
 e$="DURCHGANG BENUTZT": LET j=4
1080 NEXT j
1090 GO SUB 60: GO TO 1000
1100 LET nap=CODE i$-96
1110 LET m$(ap,j)=CHR$ 42
1120 FOR j=1 TO 4
1130 IF m$(nap,j)=CHR$ (ap+64) T
HEN LET m$(nap,j)=CHR$ 42

1140 NEXT j
1150 IF nap=dp THEN LET w$="ANGR
EIFER"
1160 LET x=ap: LET y=nap: IF nap
>ap THEN LET x=nap: LET y=ap
1170 PLOT q(x,1),q(x,2)
1180 DRAW OVER 1; INK 4;q(y,1)-q
(x,1),q(y,2)-q(x,2)
1190 PRINT AT p(ap,1),p(ap,2);CH
R$ (ap+64)
1200 LET ap=nap
1210 GO SUB 70
1220 IF INKEY$=i$ THEN GO TO 122
0
1230 IF w$="" THEN GO TO 520
1240 GO TO 7000
2000 PRINT AT 21,0;"Verteidiger
zieht (";  INK 1;CHR$ 145;  INK 0;"
)       "
2010 LET i$=INKEY$
2020 IF i$="" OR i$=CHR$ 13 THEN
 GO TO 2010
2030 IF i$="a" OR i$="b" OR i$="
d" THEN GO TO 2070
2040 LET e$="BLEIBE IM BEREICH!"
2050 GO SUB 60
2060 GO TO 2000
2070 LET ndp=CODE i$-96
2080 IF ndp=dp THEN GO TO 2010
2090 IF ndp=ap THEN LET w$="VERT
```

```
2100 PRINT AT p(dp,1),p(dp,2);CH
R$ (dp+64)
2110 LET dp=ndp: GO SUB 75
2120 IF INKEY$=i$ THEN GO TO 212
0
2130 IF w$="" THEN GO TO 500
2140 GO TO 7000
3000 PRINT '"A:Du bist der ANGRE
IFER"''"D:Du bist der VERTEIDIGE
R"
3010 LET i$=INKEY$: IF i$<>"a" A
ND i$<>"d" THEN GO TO 3010
3020 LET no=CODE i$: RETURN
7000 IF w$(1)="D" AND no=100 THE
N LET w$="Du hast"
7010 IF no=97 OR (no=100 AND w$(
1)="A") THEN LET w$="Ich habe"
7020 PRINT AT 21,0;w$;" gewonnen
";: IF no=2 THEN PRINT "S";
7030 PRINT "                ": REM
10 spaces
7040 INPUT "P= Neues Spiel ;S= S
top ";i$
7050 IF i$<>"p" AND i$<>"s" THEN
 GO TO 7040
7060 IF i$="s" THEN GO TO 9999
7070 GO TO 150
8000 LET i$=CHR$ (c(ap)+64+(c(ap
)=3))
8010 PRINT AT 21,0;"MEIN ZUG ";i
$;" (Druecke ENTER)"
8020 IF INKEY$<>CHR$ 13 THEN GO
TO 8020
8030 PRINT AT 21,17;"        ": RE
M 6 spaces
8040 LET i$=CHR$ (CODE i$+32)
8050 GO TO 1050+(1020 AND no=97)
8500 LET g=g+1: IF g=1 THEN LET
i$=m$(ap,INT (RND*4)+1): GO TO 8
010
8510 DIM g(4): FOR j=1 TO 4
8520 LET pm=CODE m$(ap,j)-64
8530 IF pm=-22 THEN LET g(j)=-30
: GO TO 8600
8540 IF c(pm)=c(dp) THEN LET g(j
)=15: GO TO 8600
8550 IF pm=dp THEN LET g(j)=30:
GO TO 8600
8560 IF pm<3 OR pm=4 THEN LET g(
j)=-10: GO TO 8600
8570 FOR k=1 TO 4
8580 IF m$(pm,k)=CHR$ 42 THEN LE
T g(j)=g(j)-1
8590 NEXT k
8600 NEXT j
8610 LET k=g(1)
8620 FOR j=2 TO 4: IF g(j)>k THE
```

```
8640 IF K=-30 THEN LET w$="UERTE
IDIGER": GO TO 7000
8650 FOR j=1 TO 4: IF g(j)<>K TH
EN NEXT j
8660 LET i$=m$(ap,j): GO TO 8010
9000 SAVE "FESTUNG" LINE 0
9500 DATA 60,60,60,126,255,255,6
0,60,102,255,255,126,60,60,24,24
9999 STOP
```

ANMERKUNGEN ZUM PROGRAMM

Die durch "FORT-DATA" aufgebauten Arrays sind folgende:

p(18,2) : DATA Zeile 1000: Die Koordinaten auf dem Schirm für die 18 Räume des Schlosses.
q(18,2) Die gleichen Bildschirmpositionen, umgewandelt in PLOT-Koordinaten. (Array p wird benötigt, um die Buchstaben der Zimmerkennzeichnung und die Position der Spieler anzuzeigen. Mit Hilfe von Array q werden die Verbindungswege gezeichnet (DRAW)).
c(18) : DATA Zeile 2000: das können Sie selbst herausfinden!
I$(18,4) : DATA Zeile 3000: die "erlaubten Züge" von jedem Raum zu Beginn des Spiels. Dieses Array wird später in ein anderes übertragen (m$(18,4)), und wenn Durchgänge benutzt wurden, werden Elemente von m$ mit Sternchen (CHR$ 42) überschrieben. I$ bleibt unverändert und bereit fürs nächste Spiel.

"FESTUNG" beginnt (bei Zeile 100) mit der Definition von zwei Graphikzeichen und zwar CHR$ 144 für den Angreifer und CHR$ 145 für den Verteidiger. Die davor liegenden Zeilen enthalten vier Unterprogramme zum Zeichnen des 'Spielfeldes' (5 bis 50), Ausgabe einer Fehlermeldung (60), Ausdrucken des Angreifers (70) und Ausdrucken des Verteidigers (75).

Die vier Hauptteile des Programms sind:

1000–1240: Zug Angreifer (Eingabe durch menschlichen Spieler)
2000–2140: Zug Verteidiger (Eingabe durch menschlichen Spieler)
8000–8050: Zug Verteidiger (Computer)
8500–8650: Zug Angreifer (Computer)

Die vier Zeilen 500 bis 530 kontrollieren die Reihenfolge, in der diese Programmteile, (nur zwei davon werden in jedem Spiel benötigt), aufgerufen werden.

MINENFELD

Das ist ein scheinbar einfaches Spiel, das nicht die gleiche geistige Gewandtheit wie "Festung" erfordert, aber trotzdem ein gewisses Maß an Geschicklichkeit verlangt.

Es soll versucht werden, mit drei Männern ein Minenfeld zu überqueren, je 27 Schritte, wobei die jeweilige Zuglänge durch die Augenzahl eines simulierten Würfels festgelegt wird.

Die Minen sind alle deutlich gekennzeichnet (mit roter Farbe; sicherer Boden ist grün), und außer den beiden nebeneinander liegenden Minen nahe dem Ende, gleich weit voneinander entfernt.

Natürlich spielt das Glück eine Rolle, wie in jedem Würfelspiel, aber es erfordert Geschicklichkeit, den richtigen Mann zum Ziehen auszusuchen, um das Beste aus dem jeweiligen 'Wurf' zu machen. Stellungsspiel ist alles, sowohl im Minenfeld selbst als auch beim Endspiel, wenn Sie bereits einen oder zwei Männer hinter die beiden letzten Minen gebracht haben.

ANMERKUNGEN ZUM PROGRAMM

Array a( ) ist ein zweidimensionales Array, wobei die erste Dimension die Position von jedem der 3 Männer festhält, während die zweite ein Kennzeichen enthält, sobald einer der Männer tot ist.

Dadurch werden Versuche, einen 'toten' Mann oder einen Mann, der das Minenfeld bereits überquert hat, zu bewegen, als unerlaubte Züge erkannt (Zeile 150).

Zeile 190 benutzt ebenfalls die zweite Ebene als Parameter für FLASH, d.h., das Zeichen für einen lebenden Mann wird mit FLASH Ø, das Zeichen für einen toten mit FLASH 1 aus-

Ich habe in diesem Programm keine selbstdefinierten Graphikzeichen verwandt, also lesen Sie "Sternchen" statt "Mann". Es bleibt auch sehr viel Raum für Sie, um das Programm mit Ton, Graphik, realistischen Explosionen, Hintergrund usw. auszuschmücken.

```
  10 BORDER 6: PAPER 6: INK 0: C
LS
  20 DIM c$(32): DIM a(3,2): FOR
 j=1 TO 3: LET a(j,1)=1: NEXT j
  30 PRINT AT 4,1;: FOR j=1 TO 6
: PRINT INK 4;"---"; INK 2;"-";:
 NEXT j
  40 PRINT INK 2;"-"; INK 4;"--"
; INK 7;CHR$ 143
  50 FOR j=1 TO 3: PRINT AT j,1;
"*": NEXT j
  60 RANDOMIZE
 100 LET x=INT (RND*6)+1
 110 PRINT AT 10,0;"GEHE ";x;" S
CHRITT";: IF x<>1 THEN PRINT "E"
 120 PRINT AT 15,0;"Welcher Mann
 (1-3)"
 130 LET i$=INKEY$: IF i$<"1" OR
 i$>"3" THEN GO TO 130
 140 LET m=VAL i$
 150 IF a(m,2) OR a(m,1)=28 THEN
 GO TO 500
 160 LET pn=a(m,1)+x
 170 LET p=pn/4: IF pn<28 AND p=
INT p OR pn=25 THEN LET a(m,2)=1
 180 IF pn>28 THEN LET pn=28
 190 PRINT AT m,a(m,1);CHR$ 32;T
AB pn; FLASH a(m,2);"*"
 200 LET a(m,1)=pn
 210 LET w=0: LET d=0: FOR j=1 T
O 3: IF a(j,2) THEN LET d=d+1
 220 IF a(j,1)=28 THEN LET w=w+1
 230 NEXT j
 240 IF d=3 THEN PRINT AT 19,10;
 FLASH 1;"ALLE MAENNER TOT!": GO
 TO 700
 250 IF w=3 THEN PRINT AT 19,10;
 FLASH 1; INK 2; PAPER 7;"GRATUL
ATION": GO TO 700
 255 LET d$="MANN": IF d<>1 THEN
 LET d$="MAENNER"
 260 IF d+w=3 THEN PRINT AT 19,2
; FLASH 1;"SPIELENDE :";d;d$;" G
ETOETET!": GO TO 700
 270 IF INKEY$=i$ THEN GO TO 270
 280 GO TO 100
 500 PRINT AT 20,10; FLASH 1;"UN
GUELTIGER ZUG": FOR j=1 TO 20: N
EXT j
```

```
= weiter : X = Stop"
 710 IF INKEY$="p" THEN RUN
 720 IF INKEY$="x" THEN STOP
 730 GO TO 710
9000 SAVE "MINENFELD" LINE 0
```

NICHT NUR EIN SPIELZEUG

RECHNER
TEXT-VERARBEITUNG
FAKTURIERUNG
VERKAUFS-BUCH
BIORHYTHMUS

RECHNER

Mit Hilfe dieses Programms funktioniert Ihr Spectrum FAST so gut wie ein billiger Taschenrechner!

(Der Spectrum besitzt eine Menge eingebaute, komplexe mathematische Funktionen, sowie eine sehr große Zahl von Speicherplätzen, von denen in diesem Programm kein Gebrauch gemacht wird. Wenn Sie also vorhaben, es auszubauen, um diese Funktionen usw. verwenden zu können - bitte . . .).

BENUTZUNGSVORSCHRIFTEN

Das Programm verwendet die gleiche Logik wie die meisten Taschenrechner, d.h., Sie geben Rechenoperationen genauso ein, wie Sie sie auf Papier schreiben würden, und der Bildschirm zeigt entweder Ihre laufende Eingabe (nachdem Sie eine numerische Taste gedrückt haben) oder das bisherige Ergebnis (nach einem arithmetischen Operator). Sie hören außerdem ein irritierendes BEEP bei jedem Tastendruck (Warum haben das billige Rechner so an sich?).

Die einzigen arithmetischen Operatoren, die das Programm akzeptiert, sind +, -, *, / und =. Sie brauchen weder SYMBOL SHIFT noch den Dezimalpunkt (obwohl es nichts ausmacht, wenn Sie sie benutzen). Außerdem: Nachdem Sie vermutlich öfter einen Computer als einen Taschenrechner verwenden und daher die ENTER-Taste leichter mit dem Endergebnis in Verbindung bringen als die '= - Taste', haben diese beiden Tasten jeweils den gleichen

Effekt, nämlich die Ausgabe des Ergebnisses.

Um falsche Eingaben zu korrigieren, kann man entweder die "C"-Taste (wie unten beschrieben) oder DELETE, das normal funktioniert, verwenden. (Nicht viele Taschen-

löscht werden, aber alle Computer. Ich glaube nicht, daß Sie darauf gerne verzichten möchten.)

Die "C"-Taste hat eine Doppelfunktion: CLEAR (wie beim Taschenrechner, nicht wie beim Computer). Ohne SHIFT funktioniert es als EINGABE-LÖSCH-Taste, wobei das laufende Zwischenergebnis erhalten bleibt, und mit SHIFT (unter Verwendung von CAPS SHIFT) wird sowohl die Eingabe als auch der Speicher gelöscht.

Überlauffehler bei der Eingabe werden erkannt und angezeigt, aber Überlauffehler bei den Rechenoperationen bringen natürlich das Programm mit Fehleranzeige 6 zum Absturz.

Zeile 2010 bedarf einiger Erklärungen. a$ enthält die laufende Eingabe. Drücken Sie einen arithmetischen Operator, ohne daß eine aktuelle Eingabe vorhanden ist, dann wird die Operation mit dem Zwischenergebnis (r$) durchgeführt. Existiert auch kein Zwischenergebnis, dann wird das vorhergegangene Endergebnis (q$) verwendet.

Nur wenn weder eine Eingabe, noch ein Zwischenergebnis oder ein Endergebnis von einer vorangegangenen Operation vorhanden ist, dann wird die Operation zurückgewiesen. (Verschiedene Taschenrechner arbeiten in diesem Punkt anders, aber meiner ist ein sehr billiges Modell und funktioniert genau auf diese Weise.)

```
    1 REM RECHNER
   10 CLS : PRINT TAB 10;"ZX RECH
NER"
  100 LET r$=""
  150 LET q$=""
  200 LET s$="mkjbvl"+CHR$ 13
  250 LET x$=".+-*/"+CHR$ 13+CHR$
  13
 1000 LET a$="": LET j=0: LET d=0
 1010 PAUSE 0: LET i$=INKEY$
 1020 FOR k=1 TO 7
 1030 IF i$=s$(k) OR i$=x$(k) THE
N GO TO 2000
 1040 NEXT k
 1050 IF i$="c" OR i$="C" THEN GO
  TO 5020
```

```
 TO 1100
1070 IF i$<"0" OR i$>"9" THEN GO
 TO 1010
1080 IF j=31 THEN GO TO 5000
1090 LET a$=a$+i$: LET j=j+1
1100 GO SUB 4000: PRINT a$: BEEP
 .01,30
1110 GO TO 1010
2000 LET i$=x$(k): IF i$="." THE
N GO TO 3000
2005 IF a$="." THEN GO TO 1010
2010 IF a$="" THEN LET a$=r$: IF
 a$="" THEN LET a$=q$: IF a$=""
THEN GO TO 1000
2020 IF r$="" THEN LET r$=a$: LE
T b$=i$: PRINT AT 12,31;b$: BEEP
 .01,50: GO TO 1000
2030 LET r$=STR$ (VAL (r$+b$+a$)
)
2040 LET j=LEN r$: GO SUB 4000:
PRINT r$: BEEP .05,40
2050 LET b$=i$: IF i$=CHR$ 13 TH
EN LET b$=CHR$ 32: LET q$=r$: LE
T r$=""
2060 PRINT AT 12,31;b$: IF b$<>C
HR$ 32 THEN BEEP .01,50
2070 GO TO 1000
3000 IF d THEN GO TO 1010
3010 LET d=1: GO TO 1090
4000 DIM c$(32-j): PRINT AT 11,0
;c$;: RETURN
5000 PRINT AT 15,14; FLASH 1;"FE
HLER"
5005 LET i$=INKEY$
5010 IF i$<>"c" AND i$<>"C" THEN
 GO TO 5005
5015 PRINT AT 15,14;"      "
5020 DIM c$(32): PRINT AT 11,0;c
$
5025 IF i$="C" THEN GO TO 100
5030 GO TO 1000
9000 SAVE "RECHNER" LINE 0
```

TEXTVERARBEITUNG

Wenn Sie über einen Schönschreib-Drucker für die Ausgabe und ZX Mikro-Laufwerke oder konventionelle Disketten für die Speicherung verfügen, dann könnten Sie der Meinung sein, daß dieses Programm nicht allen Anforderungen, die an ein Textverarbeitungsprogramm gestellt werden, genügt. Wenn Sie jedoch nur eine 'Minimumkonfiguration', bestehend aus 16K-Spektrum, ZX Printer und Kassettenrecorder haben, dann sollte es mehr als ausreichend sein.

Es verfügt über die Standardmöglichkeiten eines Text-Verarbeitungsprogramms: Eingabe und Änderungsmöglichkeiten (obgleich etwas eingeschränkt - siehe unten), Speichermöglichkeit auf Band und die Möglichkeit, den Text neu zu laden, wahlweise Rechtsausrichtung (d.h. nach dem rechten Blattrand bündig ausrichten), usw.

Das Programm faßt bis zu fünf Bildschirmseiten Text (à 20 Zeilen), d.h. 100 Zeilen insgesamt. Wenn Sie mehr als 16K RAM haben, kann das erheblich erweitert werden durch Änderung des DIM Befehls in Zeile 5 (w$) und entsprechenden Änderungen in den Zeilen 1200 und 7030 (siehe Programmbeschreibung).

BENUTZUNGSANWEISUNG

EINGABE (über Tastatur)

Wenn das Programm gestartet ist, wählen Sie Möglichkeit "K" um Text über die Tastatur eingeben zu können.

Ein blinkender Cursor erscheint in der linken oberen Ecke des Bildschirms. Geben Sie Text ein, als ob Sie Schreib-

erreichen, wird ein BEEP ertönen und Sie auf das nahe Zeilenende hinweisen. Sie können weitertippen, und ein Leerraum oder Bindestrich bringt den Cursor automatisch auf die nächste Zeile. Wenn Sie weder Leerraum noch Bindestrich vor Spalte 32 benutzen, wird automatisch ein Bindestrich eingefügt. (Manchmal ist es wünschenswert und manchmal nicht. Sie könnten es vorziehen, zurückzugehen und entweder das gesamte Wort löschen, oder den Bindestrich woanders einfügen. Beides ist möglich – siehe EDITIEREN).

Wenn Sie einen neuen Absatz beginnen wollen, so können Sie den Cursor mit Hilfe der ENTER-Taste auf die nächste Zeile bringen. (Drücken Sie zweimal ENTER, wenn Sie eine Leerzeile zwischen zwei Absätzen einfügen wollen.) Sie können den neuen Absatz mit einem Leerraum oder mit mehreren beginnen, aber wenn Sie den endgültigen Text rechts ausgerichtet haben wollen, dann verwenden Sie **NUR EINEN LEERRAUM**.

Wenn Sie keine 20 Textzeilen einzugeben haben, dann geben Sie "." ein, wenn der Cursor am **LINKEN** Rand steht. (d.h. beginnen Sie eine neue Zeile mit einem Punkt.)

Wenn Sie das tun, oder wenn Sie 20 Zeilen Text eingegeben haben, erscheint folgende Meldung:

P = PRINT :J = JUSTIFY :C = CONT
Diese Möglichkeiten funktionieren auf folgende Weise:

PRINT: Erzeugt eine identische Kopie Ihres Textes auf dem ZX Printer. Anschließend erscheint die gleiche Meldung wieder, so daß Sie so viele Kopien ausdrucken können wie Sie wollen.

JUSTIFY: Richtet Ihren Text auf dem Bildschirm aus.

Außerdem wird der Text im Speicher des Computers ausgerichtet. Rückkehr zu den 3 Auswahlmöglichkeiten, so daß Sie entweder den ausgerichteten Text ausdrucken,

CONT: Bietet entweder die Möglichkeit, weiter Text einzugeben oder den gesamten Text auf Band zu speichern.

EDITIEREN

Es gibt keinen separaten 'edit-Modus', wie bei einigen Textverarbeitungsprogrammen. Sie können den Text editieren, während Sie ihn zusammenstellen (und zu keinem anderen Zeitpunkt). Überzeugen Sie sich, daß das Editieren beendet ist, bevor Sie das Ende einer Seite erreichen, denn es gibt keine Möglichkeit jemals wieder zu diesem Punkt zurückzukommen. Löschen Sie Fehler, die Sie sofort bemerken, mit DELETE wie üblich, sonst gehen Sie mit der 'Cursor-Links'-Taste (CAPS SHIFT "5") mit dem Cursor an die entsprechende Stelle und überschreiben einfach den zu ändernden Text. Führen Sie anschließend den Cursor mit 'Cursor-Rechts' (CAPS SHIFT "8") wieder zu der aktuellen Textposition.

WICHTIG

Wenn Sie eine Zeile oben auf dem Bildschirm geändert haben, könnten Sie versucht sein, den Cursor durch Drücken der ENTER-Taste schneller zum Ende der Seite zu bringen als durch CAPS SHIFT "8". Denken Sie jedoch daran, daß wenn Sie die JUSTIFY Funktion wählen, der Computer alle Zeilen, die automatisch beendet wurden, ausrichtet. Zeilen jedoch, die durch ENTER beendet wurden (was normalerweise Ende des Absatzes oder Ende des Textes bedeutet) **nicht**. Wenn Sie also diese Methode der Cursorbewegung wählen, dann sagen Sie damit dem Computer, daß keine dieser Zeilen ausgerichtet werden soll.

SPEICHERN AUF BAND

Der Beginn davon wurde bereits weiter oben erklärt. Sie werden aufgefordert, einen Dateinamen für Ihren Text anzugeben, und der Computer bestätigt, daß der Text tatsächlich unter diesem Dateinamen abgespeichert wurde.

10 Zeichen; wenn Sie also einen längeren Namen eingeben, so wird er abgeschnitten, und der Name, der auf dem Bildschirm erscheint, ist der tatsächliche Name unter dem Ihr Text gespeichert ist, und unter dem er später wieder aufgerufen werden kann).

Das Speichern des Textes beendet das Programm. Sie können das korrekte Abspeichern noch durch Eingabe von
VERIFY n$ DATA w$() überprüfen.

Sollte VERIFY nicht funktionieren, oder falls Sie aus irgend einem Grund eine weitere Kopie benötigen, geben Sie

GO TO 8030 ein.

LADEN DES TEXTES VOM BAND

Starten Sie das Programm und drücken Sie "T". Sie werden nach dem entsprechenden Dateinamen gefragt, und der Text wird geladen und direkt über den ZX-Printer ausgedruckt. Es wird jeweils eine Seite gedruckt, so daß Sie die Möglichkeit haben, einzelne Seiten zu wiederholen oder weiterzumachen, bis der Text komplett ist, wobei Sie danach wiederum wählen können, ob sie einen Teil oder den gesamten Text wiederholen wollen oder ob das Programm beendet werden soll.

ANMERKUNGEN ZUM PROGRAMM

Falls Sie weitere Teile an das Textverarbeitungsprogramm anfügen wollen, hier sind die wichtigsten Abschnitte des Programms:

300– 600: Eingabe (Zeilen 320–400 = Kontrollzeichen)
1000–1210: Druckausgabe und verschiedene Untermenues
5000–5300: Ausrichtung
7000–7070: Laden vom Band
8000–8050: Speichern auf Band

Die Variable x zeigt die 'Seitennummer' an (wobei die erste Seite die Nummer Ø trägt). Wenn Sie also die maximale Dateigröße erhöhen (bei mehr als 16k), dann achten Sie bitte darauf, daß x das neue Maximum zwar erreichen, aber nicht überschreiten kann. (Zeilen 1200 und 7030).

j zeigt die aktuelle Zeile der aktuellen Seite an, l ist die aktuelle Spalte und w$ die aktuelle Textdatei.
Deshalb identifiziert der Ausdruck

w$ (x 20+j+1,l)

der mehrere Male im Programm vorkommt, ganz genau das richtige Element von w$, das der Cursorposition entspricht.

```
    1 REM TEXTVERARBEITUNG
    5 DIM w$(100,32): DIM b$(32)
   10 CLS : PRINT TAB 6;"ZX TEXTU
ERARBEITUNG";AT 5,0;"T=TEXT VOM
BAND LADEN"''"K=TEXTEINGABE UEBE
R TASTATUR"
   15 IF INKEY$="K" OR INKEY$="k"
 THEN GO TO 100
   20 IF INKEY$="t" OR INKEY$="T"
 THEN GO TO 7000
   25 GO TO 15
  100 LET x=0
  120 IF INKEY$="S" OR INKEY$="s"
  THEN GO TO 8000
  150 CLS
  200 FOR j=1 TO 19
  210 FOR l=1 TO 31
  300 PRINT AT j,l; FLASH 1; OVER
  1;CHR$ 32
  310 PAUSE 0: LET i$=INKEY$
  320 IF i$=CHR$ 8 AND l<>0 THEN
LET l=l-2: GO TO 500
  330 IF i$=CHR$ 8 AND j<>0 THEN
PRINT AT j,0;w$(x*20+j+1,1): LET
  j=j-1: LET l=30: GO TO 500
  340 IF i$=CHR$ 9 AND l<>31 THEN
  GO TO 500
  350 IF i$=CHR$ 9 AND j<>19 THEN
  PRINT AT j,31;CHR$ (CODE (w$(x*
20+j+1,32))+19*(w$(x*20+j+1,32)=
CHR$ 13)): LET j=j+1: LET l=1: G
O TO 500
  360 IF l=0 AND i$="." THEN LET
l=31: LET j=19: GO TO 540
  370 IF i$=CHR$ 12 AND l<>0 THEN
```

```
 l=l-2: GO TO 500
 380 IF i$=CHR$ 12 AND j<>0 THEN
 PRINT AT j,0;w$(x*20+j+1,1): LE
T j=j-1: LET l=31: LET w$(x*20+j
+1,l)=CHR$ 32: LET l=l-2: GO TO
500
 390 IF i$=CHR$ 13 THEN LET l=31
: GO TO 410
 400 IF i$<CHR$ 32 THEN GO TO 31
0
 410 IF l=31 AND i$<>CHR$ 32 AND
 i$<>CHR$ 13 THEN LET i$=CHR$ 45
 420 BEEP .01,20+20*(i$=CHR$ 32)
 430 LET w$(x*20+j+1,l+1)=i$
 500 PRINT AT j,0;w$(x*20+j+1)
 510 IF w$(x*20+j+1,32)=CHR$ 13
THEN PRINT AT j,31;CHR$ 32
 520 IF l=26 THEN BEEP .5,50
 530 IF l>25 AND (i$=CHR$ 32 OR
i$=CHR$ 45) THEN LET l=32
 540 NEXT l: BEEP .5,20
 600 NEXT j
1000 PRINT AT 21,0;"P=PRINT:J=AU
SRICHTEN:C=CONT"
1010 IF INKEY$="c" OR INKEY$="C"
 THEN GO TO 1100
1020 IF INKEY$="p" OR INKEY$="P"
 THEN GO TO 1050
1030 IF INKEY$="j" OR INKEY$="J"
 THEN GO SUB 5000: GO TO 1000
1040 GO TO 1010
1050 PRINT AT 21,0;b$: COPY : GO
 TO 1000
1100 CLS : PRINT "M=Mehr Text:S=
SAVE"
1110 IF INKEY$="m" OR INKEY$="M"
 THEN GO TO 1200
1120 IF INKEY$="s" OR INKEY$="S"
 THEN GO TO 8000
1130 GO TO 1110
1200 IF x=4 THEN PRINT AT 21,0;"
Kein Platz mehr,was jetzt?": PAU
SE 0: GO TO 1100
1210 LET x=x+1: GO TO 150
5000 PRINT AT 21,0;b$
5005 FOR j=x*20+1 TO x*20+20
5010 LET a$=w$(j)
5015 IF a$(32)<>CHR$ 32 THEN GO
TO 5200
5020 LET s=0: FOR l=32 TO 1 STEP
 -1
5025 IF a$(l)<>CHR$ 32 THEN GO T
O 5035
5030 LET s=s+1: NEXT l: IF s=32
THEN GO TO 5200
5035 LET d=2
5040 FOR l=2 TO 32
```

```
5050 FOR k=32 TO l+1 STEP -1: LE
T a$(k)=a$(k-1): NEXT k
5055 LET l=l+d
5060 IF a$(32)<>CHR$ 32 THEN GO
TO 5100
5090 NEXT l
5100 PRINT AT j-x*20-1,0;a$
5150 IF a$(32)=CHR$ 32 THEN LET
d=d+1: GO TO 5040
5170 LET w$(j)=a$
5200 NEXT j
5300 RETURN
7000 INPUT "Dateiname?";n$
7005 IF n$="" THEN GO TO 7000
7010 PRINT AT 15,0;"Lade das Ban
d ";n$
7015 LOAD n$ DATA w$()
7020 LET x=0
7025 FOR j=x+1 TO x+20: LPRINT w
$(j): NEXT j
7030 LET x=x+20: IF x=100 THEN G
O TO 7055
7035 CLS : PRINT "Eingabe S fuer
 Stop oder jede   andere Taste z
um weitermachen"
7040 PAUSE 0
7045 IF INKEY$="s" OR INKEY$="S"
 THEN GO TO 7055
7050 GO TO 7025
7055 CLS : PRINT "AUSGABE BEENDE
T"''"R FUER WIEDERHOLUNG"''"X FU
ER ENDE"''"EINGEBEN "
7060 IF INKEY$="R" OR INKEY$="r"
 THEN GO TO 7020
7065 IF INKEY$="X" OR INKEY$="x"
 THEN GO TO 9999
7070 GO TO 7060
8000 INPUT "bitte Dateiname eing
eben ";n$
8010 IF n$="" THEN GO TO 8000
8020 IF LEN n$>10 THEN LET n$=n$
( TO 10)
8030 PRINT "abgespeichert als: "
;n$
8040 SAVE n$ DATA w$()
8050 GO TO 9999
9000 CLEAR : SAVE "TU" LINE 0
9999 STOP
```

FAKTURIE-RUNG (48K)/ VERKAUFS-BUCH (48K)

Sollten Sie ein eigenes Geschäft besitzen, so muß ich Ihnen sicher nicht lange erklären, wie ungeheuer nützlich ein eigener Computer sein kann, um z.B. Buchhaltung, Lagerverwaltung u.ä. mit einem Minimum an Zeitaufwand zu erledigen.

Wenn nicht, dann sind Sie vielleicht versucht, die nächsten beiden Programme zu überblättern oder gleich zum nächsten Kapitel mit Spielen weiterzuspringen. Aber warten Sie eine Sekunde! Haben Sie jemals ein Programm geschrieben und Kopien davon an Ihre Freunde verkauft, oder ein Listing an eins von den Computermagazinen geschickt? Es könnte sein, daß Sie schon als 'Software-Produzent' im Geschäft sind, und wenn Sie ein unangenehmes Zusammentreffen mit dem Finanzamt vermeiden wollen, sollten Sie allmählich anfangen, Unterlagen über Ihre sämtlichen Transaktionen aufzubewahren.

Diese beiden Programme bilden zusammen die VERKAUFsseite von etwas, das gewöhnlich als Finanzbuchhaltungspaket bezeichnet wird. Damit sind mehrere Programme gemeint, die alle verschiedenen Tätigkeitsbereichen dienen, jedoch aufeinander aufbauen, die gleichen Daten benutzen und zusammen einen kompletten Satz Konten verwalten.

FAKTURIERUNG erstellt eine richtige Rechnung, die Sie

jede Transaktion in einem RECHNUNGS-AUSGANGS-BUCH fest. Es handelt sich dabei einfach um eine Liste Ihrer Verkäufe, bestehend aus dem Datum, dem Kundennamen und dem Betrag. Die Endsumme des Rechnungs-Ausgangsbuches entspricht den Gesamteinnahmen Ihres Geschäftes im entsprechenden Zeitraum.

Natürlich, wenn Sie nicht ausschließlich Barzahlungsgeschäfte abwickeln, so bedeutet die Tatsache, daß Sie Rechnungen ausgestellt haben, noch nicht, daß alle Ihre Kunden auch bezahlen - rechtzeitig und auch vollständig. Die Kontrolle der DEBITOREN ist einer der wichtigsten Teile des Geschäftes, und man muß nicht unbedingt die Größe eines multinationalen Konzerns haben, um die Kontrolle darüber zu verlieren. Und hier kommt VERKAUFS-BUCH ins Spiel.

Dieses Programm verwendet die Daten des Rechnungs-Ausgangsbuches (wie berechnet in FAKTURIERUNG und in beiden Programmen als "RECHNUNG" bezeichnet), um jedes einzelne Kundenkonto aufzulisten. Alle erstellten Rechnungen sind in einer Spalte aufgeführt, alle eingegangenen Zahlungen in einer anderen. Wenn weitere Zahlungen eingegangen sein sollten, so können Sie sie auf dem entsprechenden Kontenblatt eintragen. (Kontenblatt ist ein Prä-Computer-Ausdruck, der immer noch verwendet wird, und bedeutet die Aufzeichnung aller Bewegungen auf einem Konto). Die Endsummen der Rechnungen, Zahlungen und offenen Posten werden angezeigt. Sie können, wenn Sie wollen, 'Hard Copies' von den Kontenblättern anfertigen, sei es zu Ihrer eigenen Information oder als Mahnungen, die an die Kunden verschickt werden.

HINWEISE FÜR DIE BENUTZUNG

FAKTURIERUNG

Nachdem Sie das Programm geladen haben, erscheint als erstes ein dreiteiliges Menü auf dem Bildschirm (siehe Zeile 110). Wir betrachten diese 3 Wahlmöglichkeiten

1: Rechnung erstellen

Hierdurch wird eine Liste aller Ihrer Kunden ausgegeben, mit einer Codenummer neben jedem einzelnen. Am Ende der Liste, mit Nummer Ø, steht "NEUER KUNDE". (Wenn Sie das Programm zum ersten Mal laufen lassen, wird das der einzige Punkt auf der Liste sein). Wenn Sie bereits eine Menge Kunden haben, werden Sie die Fähigkeit des Spectrum zum 'Scrollen' einsetzen müssen, um ans Ende der Liste zu gelangen.

Geben Sie die benötigte Kundennummer ein. Nummern außerhalb des Nummernkreises werden nicht angenommen. Geben Sie Nummer "Ø" ein, dann werden Sie gebeten, die Adresse des Kunden einzugeben (insgesamt 4 Zeilen, wobei Zeile 1 natürlich den Namen des Kunden oder den seines Geschäfts enthalten sollte.)

Dann wird die Rechnung auf dem Bildschirm aufgebaut. Standardangaben, wie z.B. Ihr Name und Ihre Adresse werden automatisch ausgegeben; sie brauchen nur einzugeben:

a) Das Datum

b) Den 'Text' (d.h., für welche Waren bzw. Dienstleistungen die Rechnung erstellt wird. Fassen Sie sich kurz, weil höchstens 22 Zeilen Text gedruckt werden).

c) Den Betrag.

Wenn mehr als ein Posten auf die Rechnung gesetzt werden soll, können Sie mehrere Texte und Beträge eingeben, (aber nicht zu viele, sonst verschwindet der Rechnungskopf am oberen Bildschirmrand, bevor die Rechnung abgeschlossen ist.)

Sie können die Rechnung dann entweder mit COPY ausdrucken, oder löschen, falls Sie einen Fehler gemacht haben.

2: Rechnungen auflisten

Sie brauchen überhaupt nichts zu tun, wenn Sie diese Möglichkeit wählen. Der Computer gibt das Rechnungs-Ausgangsbuch direkt über den ZX Printer aus.

3: SAVE

Sie brauchen diese Möglichkeit nicht nach jeder Rechnung zu wählen, oder auch überhaupt nicht, wenn Sie bei dem gegenwärtigen Lauf keine Rechnung erstellt haben. Sie **MÜSSEN** sie jedoch am Ende jedes Laufes wählen, bei dem Rechnungen erstellt **WURDEN**.

Es müssen zwei Dinge abgespeichert werden:

a) Die Rechnungsliste/Rechnungsausgangsbuch, welche als Datei unter dem Dateinamen "RECHNUNG" gespeichert wird.

b) Das Programm selbst, durch den letzten Fakturierungslauf auf den neuesten Stand gebracht, bereit für den nächsten Lauf.

Wichtig: Die Wahlmöglichkeiten 1 und 2 kehren nach Beendigung zum Hauptmenü zurück, Nummer 3 beendet das Programm. Um es nach der Beendigung oder nach einem BREAK wieder zu starten, verwenden Sie bitte GO TO 100 (Nicht RUN!).

```
   10 DIM a$(100,4,20)
   20 LET nc=0: LET in=0
   30 DIM b$(32)
   40 DIM c$(100,3,10)
  100 BORDER 6: PAPER 6: INK 9: C
LS
  110 PRINT AT 5,0;"1. Rechnung e
rstellen"''''"2. Rechnungen aufli
sten"''''"3. SAVE "
  120 LET i$=INKEY$: IF i$<"1" OR
  i$>"3" THEN GO TO 120
  130 GO TO VAL i$*1000
  500 DATA "AGENTUR COOPERATION"
  510 DATA "PRINZENSTRASSE 43"
  520 DATA "8000 MUENCHEN 19"
```

```
1000 BORDER 5: PAPER 5: CLS
1010 FOR J=1 TO NC
1020 PRINT J;TAB 4;A$(J,1)
1030 NEXT J
1040 PRINT 0;TAB 4;"NEUER KUNDE"
1050 INPUT "Kundennummer? ";t
1060 IF t<0 OR t<>INT t THEN GO
TO 1050
1070 IF t<>0 THEN GO TO 1100
1080 LET nc=nc+1: FOR j=1 TO 4:
INPUT AT j*2,0;("Anschriftszeile
 ";j);": ";a$(nc,j): NEXT j
1090 LET t=nc
1100 LET in=in+1
1110 CLS : RESTORE : FOR j=0 TO
3: READ t$: PRINT AT j,(32-LEN t
$)/2;t$: NEXT j: PRINT : PRINT "
RECHNUNG No.";in: PRINT
1120 INPUT "Datum ";t$;: LET c$(
in,1)=t$: LET c$(in,2)=a$(t,1)
1130 PRINT TAB 31-LEN t$;t$
1140 FOR j=1=1 TO 4: PRINT a$(t,
j): NEXT j: PRINT '': LET ip=0
1150 INPUT "text ";t$: IF LEN t$
>22 THEN LET t$=t$( TO 22)
1160 PRINT t$;
1170 INPUT "Preis ";p: GO SUB 50
00: IF x THEN GO TO 1170
1180 PRINT TAB 23;m$: LET ip=ip+
p
1190 INPUT "weitere Positionen (
J/N)? ";t$
1200 IF t$="j" OR t$="J" THEN GO
 TO 1150
1210 IF t$="n" OR t$="N" THEN GO
 TO 1230
1220 GO TO 1190
1230 PRINT AT 20,0;"TOTAL";: LET
 P=IP: GO SUB 5000
1240 PRINT TAB 23;m$
1250 PRINT AT 21,0;"C= COPY :X=
LOESCHEN"
1260 IF INKEY$="c" OR INKEY$="C"
 THEN PRINT AT 21,0;b$: COPY : L
ET c$(in,3)=m$: GO TO 100
1270 IF INKEY$="x" OR INKEY$="X"
 THEN GO TO 1110
1280 GO TO 1260
2000 CLS : LET p=0: FOR j=1 TO i
n
2010 FOR k=1 TO 3
2020 LPRINT c$(j,k);
2030 NEXT k: LPRINT : LET p=p+VA
L c$(j,3)
2040 NEXT j
2050 LPRINT : LPRINT " TOTAL ";T
AB 20;
2060 GO SUB 5000: LPRINT m$
```

```
3000 BORDER 4: PAPER 4: CLS
3010 PRINT "SAVEn Sie zuerst die
 Rechnungs- liste fuer das VERKA
UFSBUCH Pro-gramm"
3020 SAVE "RECHNUNG" DATA C$()
3030 CLS : PRINT "SAVEn Sie jetz
t das gesamte Pro- gramm fuer di
e naechste Verwen- dung als Fakt
urierprogramm"
3040 SAVE "FAKTURA" LINE 100
3050 GO TO 9999
5000 LET x=0: DIM m$(8): LET m$=
STR$ p
5010 IF m$(8)=CHR$ 32 THEN LET m
$=CHR$ 32+m$( TO 7): GO TO 5010
5020 FOR j=1 TO 8
5030 IF m$(j)="." THEN GO TO 505
0
5040 NEXT j
5050 IF j=9 THEN LET m$=m$(4 TO
)+".00"
5060 IF j=7 THEN LET m$=m$(2 TO
)+"0"
5070 IF m$(6)<>"." THEN LET x=1
5080 RETURN
9000 CLEAR : SAVE "FAKTURA" LINE
 0
```

VERKAUFS-BUCH

Dieses Programm läuft nicht menügesteuert, sondern der Reihenfolge nach ab. Folgen Sie lediglich den Anweisungen, so wie sie erscheinen.

Laden Sie zuerst das letzte "RECHNUNG"-Band. Das erste Kontenblatt wird dann am Schirm ausgegeben. Beim ersten Mal, wenn Sie das Programm laufen lassen, bleibt die "ZAHLUNGEN"-Spalte natürlich leer.

Sie werden gefragt, ob Sie eine Zahlung verbuchen (d.h. eingeben - noch mehr prä-Computer Terminologie) wollen. Wenn ja, müssen Sie das Datum (der Zahlung) und den Betrag eintippen, und die Frage wird wiederholt, damit Sie weitere Zahlungen verbuchen können, wenn nötig. Wenn Sie fertig sind, wird die Gesamtsumme gebildet, und Sie können entweder das Kontenblatt mit dem ZX-Printer ausdrucken lassen, oder zum nächsten Konto weitergehen.

Am Ende des Laufes werden die gesamten Außenstände (alle Konten) angezeigt, und Sie werden aufgefordert, eine neue Kopie des Programms für den nächsten Lauf abzuspeichern. (Es ist nicht unbedingt nötig, wenn Sie keine neuen Zahlungen verbucht haben, aber eine gute Angewohnheit, weil man oft am Ende eines Laufes vergessen hat, ob man nun Zahlungen verbucht hat oder nicht!).

WICHTIG: Benutzen Sie immer die neueste Kopie von jedem der Programme, wenn Sie einen 'Lauf' durchführen. Das 'Hauptprogramm' selbst enthält keine Daten, aber jedesmal, wenn Sie eine Kopie davon SAVEn, dann werden die aktuellen Daten mit dem Programm zusammen abgespeichert.

```
    1 REM VERKAUFSBUCH
   10 DIM p$(100,3,10)
   20 LET np=0
   30 DIM b$(32)
   40 LET tos=0
  100 BORDER 6: PAPER 6: INK 9: C
LS
  110 PRINT "Bitte laden Sie Ihr
RECHNUNG Band"
  120 LOAD "RECHNUNG" DATA c$()
  500 FOR q=1 TO 100: CLS
  510 LET a$=c$(q,2)
  520 LET c$(q,2)="x"
  530 IF a$=b$( TO 10) THEN LET q
=100: GO TO 990
  540 IF a$="x"+b$( TO 9) THEN GO
  TO 990
  550 PRINT TAB 12;a$
  560 PRINT "DATUM";TAB 10;"RECHN
UNGEN";TAB 21;"ZAHLUNGEN"
  570 LET t=VAL c$(q,3)
  580 PRINT c$(q,1);c$(q,3)
  590 FOR k=1 TO 100
  600 IF c$(k,2)<>a$ THEN GO TO 6
40
  610 LET t=t+VAL c$(k,3)
  620 PRINT c$(k,1);c$(k,3)
  630 LET c$(k,2)="x"
  640 IF c$(k,2)="x"+b$( TO 9) TH
EN GO TO 660
  650 IF c$(k,2)=b$( TO 10) THEN
LET k=100
  660 NEXT k
  670 LET t1=0
  700 FOR k=1 TO np
  710 IF p$(k,2)<>a$ THEN GO TO 7
40
  720 LET t1=t1+VAL p$(k,3)
  730 PRINT p$(k,1);TAB 21;p$(k,3
)
  740 NEXT k
  750 INPUT "Wollen Sie eine Zahl
ung ein-      geben (J/N)";t$
  760 IF t$="n" OR t$="N" THEN GO
  TO 850
  770 IF t$="j" OR t$="J" THEN GO
  TO 790
  780 GO TO 750
  790 LET np=np+1: INPUT "Datum "
;p$(np,1): PRINT p$(np,1);
  800 INPUT "Betrag ";p: GO SUB 5
000: IF x THEN GO TO 800
  810 PRINT TAB 20;m$: LET p$(np,
3)=m$: LET p$(np,2)=a$
  820 LET t1=t1+VAL m$
  830 GO TO 750
  850 PRINT "TOTAL";TAB 10;
  860 LET p=t: GO SUB 5000
```

```
  890 PRINT m$
  900 LET p=((INT (t*100)+.1)-(IN
T (t1*100)+.1))/100: GO SUB 5000
  910 PRINT "OFFEN:";TAB 10;m$: L
ET tos=tos+VAL m$
  920 PRINT AT 21,0;"C=COPY;N=NAE
CHSTE ZAHLUNG"
  930 IF INKEY$="c" OR INKEY$="C"
  THEN PRINT AT 21,0;b$: COPY : G
O TO 920
  940 IF INKEY$="n" OR INKEY$="N"
  THEN GO TO 990
  950 GO TO 930
  990 NEXT q
1000 BORDER 4: PAPER 4: CLS
1010 LET p=tos: GO SUB 5000
1020 PRINT "GESAMT AUSSENSTAENDE
: ";m$''
1030 PRINT "Jetzt SAVEn fuer nae
chste Ver-  wendung als VERKAUFs
-Programm"
1040 SAVE "VERKAUF" LINE 30: GO
TO 9999
5000 REM Fuegen Sie hier das
CASH-Unterprogramm ein  (siehe
Text)
9000 CLEAR : SAVE "VERKAUF" LINE
  0
```

ANMERKUNGEN ZUM PROGRAMM

Beachten Sie, daß keines der beiden Programme das sehr wichtige Unterprogramm ab Zeile 5000 enthält. Keine Sorge, es kostet Sie nichts extra. Es ist ein so unglaublich nützliches Unterprogramm, das wahrscheinlich in allen Ihren Finanzprogrammen enthalten sein wird, daß wir es auf Seite 94 separat aufgelistet haben.

Falls Sie, wie jeder normale Leser, dieses Buch von hinten angefangen haben, dann haben Sie vermutlich schon eine Kopie von "CASH" auf Band. In dem Fall MERGEn Sie es einfach in beide Programme. Andernfalls SAVEn Sie, was Sie bisher eingegeben haben, geben NEW ein, tippen "CASH" ein, SAVEn es und dann laden Sie es mit MERGE in das laufende Programm. Tippen Sie das Unterprogramm nicht einfach am Ende jedes Programms ein, denn das würde eine Menge zusätzlicher Arbeit bedeuten, und außerdem hätten Sie immer noch keine separate Kopie für

Wahrscheinlich haben Sie mit diesen Programmen das Problem, daß der Fernsehschirm einfach zu klein für die Menge der Daten ist, die Sie ausgeben müssen. Rechnungen mit mehr als 6 Zeilen Text oder Kontenblätter mit mehr als 18 Transaktionen zum Beispiel.

Sollten Sie ein sehr kleines Geschäft haben, dann wird das nicht vorkommen, passiert es Ihnen aber doch, dann ist die einfachste Abhilfe, auf die Anzeige am Bildschirm zu verzichten – außer für die Menü-Auswahl und INPUT-Zeilen. Ändern Sie alle anderen PRINT Befehle in LPRINT, und erstellen Sie Rechnungen und Kontenblätter direkt über den Drucker. (Löschen Sie die COPY-Anweisungen).

Die hauptsächlich benutzten Dateien sind folgende:

a$(100,4,20) : Namen und Adreßdatei :nur für FAKTURIERUNG
p$(100,3,10) : Zahlungen: nur VERKAUFS-BUCH
c$(100,3,10) : Rechnungen: beide Programme

Die drei Elemente sowohl von c$ und p$ (2. Dimension) sind:

c$(j,1) : Datum
c$(j,2) : Kopie der Adreßzeile 1 (Name) aus der Kundendatei.
c$(j,3) : Betrag

'in' ist ein Variablenname (für Rechnungsnummer) und sollte nicht mit dem Spectrum Schlüsselwort 'IN' verwechselt werden.

Die DATA-Zeilen (500 bis 530) des Programms "FAKTURIERUNG" sollten natürlich Ihren eigenen Namen und Ihre Anschrift enthalten.

BIO-RHYTHMUS

Was, so könnten Sie fragen, tut ein Programm wie "BIO-RHYTHMUS" im Kapitel mit den ernsthaften Anwendungen, zusammen mit so ernsthaften Programmen wie 'Fakturierung' und 'Verkaufs-Buch'?

Wir wollen uns nicht mit der strittigen Frage befassen, ob das Studium der Biorhythmen eine ernsthafte Angelegenheit oder nur ein harmloser Spaß ist.

Es genügt darauf hinzuweisen, daß 'ERNSTHAFTE' Anwendung nicht unbedingt 'GESCHÄFTLICHE' Anwendung bedeuten muß, und es gibt ein weites Spektrum von Anwendungen für Ihren Spectrum, die nicht **unbedingt entweder** Geschäft **oder** Spiele sein müssen!

Das Programm zeichnet drei separate 'Biokurven' (körperliche-, seelische- und geistige Zyklen) für eine Periode von ungefähr 4 Monaten von einem beliebigen, von Ihnen zu bestimmenden Datum ab.

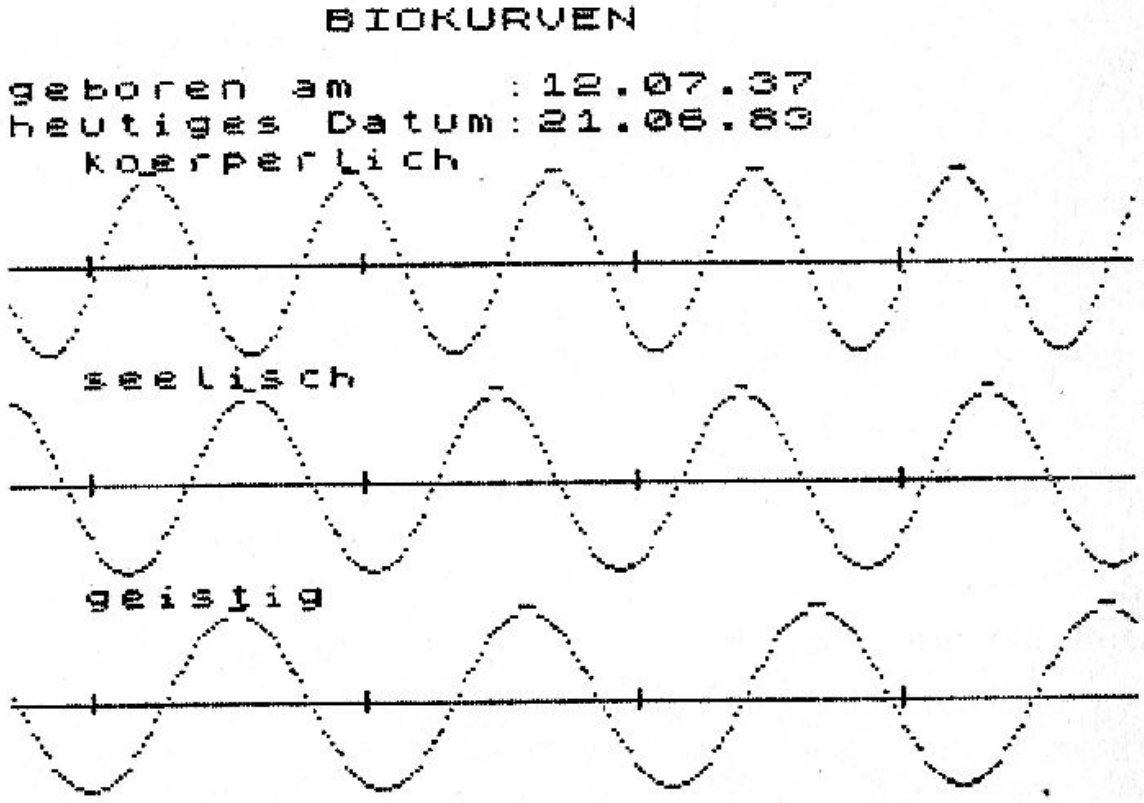
Die eigentliche Zeichenroutine ist auf die drei Zeilen 220-240 begrenzt. Der gesamte Rest befaßt sich mit solch weltlichen Dingen wie:

Eingaben durch den Benutzer :Zeilen 30-90
Zeichnen der Basislinien :Zeile 210
Einfügen der Skalierung :Zeilen 300-330. Jeder Strich auf der Basislinie zeigt den ersten eines Monats an. Daher kommt die geringe Ungleichheit der Abschnitte.

Überprüfung der Eingaben :Zeilen 1000-1100. Nicht weniger als 11 Zeilen, um sicherzugehen, daß Sie ein gültiges Datum eingegeben haben! Diese Routine kann stück-

Beispiel gestatten beide Programme, sowohl "Fakturierung" als auch "Rechnungs-Buch" jede beliebige Eingabe bei der Anfrage nach dem Datum. Sie könnten dort diese Datums-Prüfroutine einfügen, oder bei anderen Programmen.

Berechnung der Anzahl der Tage zwischen Ihrem eingegebenen Datum und einem hypothetischen Basisjahr (Jahr Ø): Zeilen 1500–1550. Das eigentliche Basisdatum ist unwichtig, da das Programm lediglich die Anzahl der Tage zwischen den beiden eingegebenen Daten berechnen muß.




```
    1 REM Biorhythmus
   10 BORDER 6: PAPER 6: INK 9: B
RIGHT 1: CLS
   20 LET m$="303232332323"
   30 PRINT TAB 11;"BIORHYTHMUS"'
''''"Bitte Dateneingabe wie folg
t:"''''TAB 12;"DD.MM.JJ"
   40 INPUT "geben Sie Ihren Gebu
rtstag ein";d$
   50 GO SUB 1000: IF x THEN GO T
O 40
```

```
 eingeben ";d$
  80 GO SUB 1000: IF x THEN GO T
O 70
  90 LET n=n-b: IF n<0 THEN GO S
UB 2000: GO TO 70
 100 DIM n$(3,12): LET n$(1)="ko
erperlich": LET n$(2)="seelisch"
: LET n$(3)="geistig"
 110 DIM b(3): DIM c(3)
 120 FOR j=1 TO 3
 130 LET b(j)=18+5*j
 140 LET c(j)=n-b(j)*INT (n/b(j)
)
 150 NEXT j
 200 CLS : PRINT TAB 9;"BIOKURVE
N"''"geboren am      :";b$,"heutig
es Datum:";d$
 210 FOR j=1 TO 3: PLOT 0,j*48-2
8: DRAW 255,0: PRINT INK j;AT j*
6-2,2;n$(j): NEXT j
 220 FOR j=1 TO 3: FOR k=0 TO 25
5
 230 PLOT INK j;k,SIN ((k+(2*c(j
)))*PI/b(j))*20+((4-j)*48-28)
 240 NEXT k: NEXT j
 300 FOR j=0 TO 255
 310 LET d=d+.5
 320 IF d>CODE m$(m)-20 THEN LET
 d=.5: FOR k=1 TO 3: PLOT j,k*48
-30: DRAW 0,5: NEXT k: LET m=m+1
: IF m>12 THEN LET m=1
 330 NEXT j
 340 STOP
1000 LET x=0: PRINT AT 15,10;"
            ": REM 12 Leerraeume
1010 IF LEN d$<>8 THEN GO TO 200
0
1020 IF d$(3)<>"." OR d$(6)<>"."
 THEN GO TO 2000
1030 FOR j=1 TO 7 STEP 3
1040 IF d$(j)<"0" OR d$(j)>"9" O
R d$(j+1)<"0" OR d$(j+1)>"9" THE
N GO TO 2000
1050 NEXT j
1060 LET y=VAL d$(7 TO ): LET m=
VAL d$(4 TO 5): LET d=VAL d$( TO
 2)
1070 IF y=0 OR d=0 OR m=0 OR m>1
2 THEN GO TO 2000
1080 LET l=0: IF y/4=INT y/4 THE
N LET l=1
1090 LET n=CODE m$(m)-20+(m=2 AN
D l=1)
1100 IF d>n THEN GO TO 2000
1500 LET n=0
1510 FOR j=1 TO m-1
1520 LET n=n+CODE m$(j)-20
1530 NEXT j
```

```
1550 LET n=n+d+INT (365.25*(y-1)
): RETURN
2000 LET x=1
2010 PRINT INK 2; FLASH 1;AT 15,
10;"ungueltiges Datum";y
9000 SAVE "BIO" LINE 0
```

DAS GEDÄCHTNIS

READER 1
READER 2
TRACE
RENUMBER
GRÖSSEN-WECHSEL
VARIABLE

DAS GEDÄCHTNIS

Die Programme in diesem Kapitel möchten einen Teil des RAM im Speicher des Spectrums untersuchen.

Sie werden aus Kapitel 24 des Spectrum-Handbuches wissen, daß das RAM weit davon entfernt ist, leer zu sein, sogar dann, wenn der Computer zum erstenmal eingeschaltet wird. Wenn Sie einen 16K Spectrum besitzen, sind Sie vielleicht entsetzt, wenn Sie entdecken, daß rund 6K durch den Bildschirmspeicher und ein weiteres K durch den Attributspeicher belegt sind und Ihnen somit gerade noch 9K für Ihre Programme bleiben.

Der Bereich des RAM jedoch, auf den ich mich konzentrieren möchte, **IST** frei, wenn Sie den Rechner einschalten, und zwar handelt es sich um den Bereich zwischen PROG und E-LINE, d.h. der Bereich, in dem Ihr Programm und die Variablen abgespeichert werden.

Um die Grenzen dieses Bereichs zu finden, geben Sie die folgenden drei Zeilen als DIREKTBEFEHLE (nicht als Programm) ein.

Um PROG zu finden, die Adresse zu Beginn des Programmbereichs:

```
PRINT PEEK 23635 + 256 PEEK + 23636
```

Um VARS, die Adresse zwischen dem Programm- und dem Variablenbereich zu finden:

```
PRINT PEEK 23627 + 256 PEEK + 23628
```

Um E-LINE, die Endadresse des Variablenbereichs zu finden:

PRINT PEEK 23641 + 256 PEEK + 23642

Sie werden feststellen, daß die beiden ersten Befehle zum gleichen Ergebnis führen, und daß der dritte ein Resultat erzielt, das nur um 1 höher ist (weil der Variablenbereich **immer** ein Byte 80h (CHR$ 118) enthält, das als Endkennzeichen für diesen Bereich dient).

Das würde bedeuten, daß das Programm und seine Variablen insgesamt nur 1 Byte Speicherplatz belegen. Eigentlich nicht erstaunlich, denn zu diesem Zeitpunkt haben Sie überhaupt noch kein Programm eingegeben!

Jetzt geben Sie die gleichen drei Zeilen ein, diesés Mal jedoch als Programm (d.h. mit Zeilennummern, sagen wir 10, 20 und 30) und dann RUN.

Dieses Mal bleibt zwar PROG gleich, aber VARS und E-LINE liegen beide um 123 Bytes höher, das bedeutet, daß Ihr kurzes Drei-Zeilen-Programm 123 Bytes im RAM belegt. E-LINE liegt immer noch um nur ein Byte oberhalb von VARS, denn Ihr Programm verwendet keine Variablen.

Schließlich ändern Sie die drei Programmzeilen noch, indem Sie PRINT durch LET a=, LET b= und LET c= ersetzen und eine vierte Zeile,

40 PRINT a'b'c

eingeben. Dadurch wird die Programmlänge auf 140 Bytes erhöht und der Variablenbereich auf 13 Bytes.

Offensichtlich enthält der Variablenbereich jetzt Werte für a, b und c, und diese belegen die 13 Bytes, so werden Sie jetzt denken. **Total falsch!**

Schauen Sie noch einmal auf die dritte Zeile Ihres Pro-

(PEEK die Systemvariable E-LINE) wird **zuerst** bestimmt, und dann wird die Variable c auf den gleichen Wert gesetzt. Deshalb finden Sie zu diesem Zeitpunkt das Ende des Variablenbereichs, der nur a enthält, b und c wurden noch nicht bestimmt.

Starten Sie das Programm erneut, und wenn es beendet ist, geben Sie GO TO Ø ein (um es wieder zu starten ohne die Variablen zu löschen). Jetzt werden Sie die echte Adresse von E-LINE erhalten, 6 Bytes höher als zuvor. Der Grund dafür ist folgender: Der Variablenbereich belegt insgesamt 19 Bytes, pro Variable 6, und eines für CHR$ 118.

Nun, jetzt wissen Sie, WO der Spectrum Ihr Programm und die Variablen abspeichert – geben wir einige richtige Programme ein und finden heraus, WAS er genau speichert.

READER 1

Wir wissen, wo im Speicher des Spectrum ein Programm gespeichert ist, im Bereich zwischen PROG und VARS, und wir können die echten Adressen von PROG und VARS finden, indem wir die entsprechenden Systemvariablen ein-PEEKen. Wenn wir also alle Adressen zwischen PROG und VARS PEEKen, dann sollten wir in der Lage sein, unser Programm zu finden.

READER 1 ist ein allgemeines Allzweck-'PEEK-Programm', das, wenn ihm eine Start- und Endadresse gegeben wird (in Zeilen 10 und 15), alle Adressen dazwischen liest. Nun ist der Programmbereich ein logischer Ort, um mit PEEK zu lesen, weil wir zumindestens wissen, was wir hier zu finden **erwarten**. Sie würden doch nach allem, was Sie bisher erfahren haben, erwarten, daß das Programm in genau der gleichen Weise gespeichert ist, wie es eingegeben wurde. (Oder etwa nicht?)

Starten Sie das Programm und schauen Sie, was passiert. (Alle Programme in diesem Kapitel arbeiten mit sich selbst, außer RENUMBER, oder mit irgend einem anderen Programm, das gleichzeitig mit ihnen im Speicher ist. Wenn es Ihnen also langweilig wird, READER 1 zu 'lesen', können Sie ein anderes Programm mit MERGE dazuladen und sich auch durch dieses lesen. Achtung, daß keine Zeilen miteinander kollidieren.)

Die Ausgabe des Programms erfolgt in drei Spalten: Spalte 1 listet jede Adresse zwischen PROG und VARS auf, Spalte 2 gibt den 'gelesenen' Inhalt dieser Adressen wieder, und Spalte 3 gibt das Zeichen an, das durch die gePEEKten Werte repräsentiert wird.

Für jeden anderen Bereich des Speichers wäre Spalte 3 nicht sehr informativ, aber was den Programmbereich betrifft, so sollte in Spalte 3 das Programm genauso erschei-

Natürlich geschieht das nicht. Aber obwohl es einige geheimnisvolle Zeichen und ein oder zwei 'nicht abdruckbare' Dinge gibt (das sind keine schmutzigen Wörter, sondern Farb-Kontrollzeichen, die eine Fehlermeldung zur Folge hätten, wenn man sie zu drucken versuchte), so bleibt doch genug Gleiches zwischen Spalte 3 und Ihrem Originalprogramm, daß es klar zu erkennen ist. Einiges davon wurde genau in der gleichen Weise gespeichert wie Sie es eingetippt haben, aber andere Dinge wurden auf recht seltsame Art gespeichert. Versuchen Sie herauszufinden, was passiert ist, und erklären Sie die seltsamen Zeichen, bevor Sie READER 2 eingeben.

```
   10 LET p=PEEK 23635+256*PEEK 2
3636
   15 LET v=PEEK 23627+256*PEEK 2
3628
   20 LET c=PEEK p
   25 PRINT p;TAB 8;: IF c<16 OR
c>23 THEN PRINT TAB 16;CHR$ c: G
O TO 35
   30 PRINT TAB 16;"(undruckbar)"
   35 LET p=p+1: IF p=v THEN STOP

   40 GO TO 20
9000 SAVE "READER 1" LINE 0
```

READER 2

Das ist eine verbesserte Version des vorhergehenden Programmes, bei der Ihnen Spalte 3 das genaue Programmlisting wiedergibt. (Es könnten immer noch einige nicht abdruckbare Zeichen erscheinen, falls Sie Farb-Kontrollzeichen benutzen, um Ihr Programm in prächtigen Farben aufzulisten.)

"Reader 3", falls Sie das für sich selbst schreiben wollen, könnte dies ebenfalls erklären.

Wie Sie bei READER 1 gesehen haben werden, besteht der Hauptunterschied zwischen einem Programm im Speicher und einem aufgelisteten Programm in der Art, wie Zahlen gespeichert werden. Der Spectrum unterscheidet zwischen Zeilennummern und allen anderen Zahlen, und speichert sie auf völlig verschiedene Art.

ZEILENNUMMERN

Zeilennummern (d.h. Zahlen, die vor jeder Programmzeile stehen. Zeilennummern, die einem GO TO oder GOSUB folgen, werden auf die gleiche Weise wie andere Zahlen gespeichert).

Zeilennummern werden als zwei Bytes große Hexadezimalzahlen gespeichert. Zahlen, die weniger als 2 Bytes benötigen, d.h. Zahlen unter 256 dezimal, werden mit einer führenden Null gespeichert.

Deshalb besteht die erste Änderung von READER 2 darin, daß, wann immer es sich um eine Zeilennummer handelt, 2 Bytes aus dem Speicher in eine einfache Dezimalzahl umgewandelt werden (Zeilen 20 und 25).

Die beiden Bytes, die der Zeilennummer folgen, sind KEIN Teil des Programms. Sie werden durch den Computer ein-

zeigen, und dienen als ZEIGER auf die nächste Adresse, die als Zeilennummer behandelt wird.

Auch diese beiden Bytes stellen eine Hexadezimalzahl dar, aber sie sind in umgekehrter Reihenfolge gespeichert. Das 'HIGHT ORDER'-Byte, (d.h. dasjenige, dessen Wert mit 256 multipliziert werden muß, wenn man es in eine Dezimalzahl umwandelt) ist **hinter** dem 'LOW ORDER'-Byte abgespeichert. Das klingt vielleicht unlogisch, aber Sie werden feststellen, wenn Sie tiefer in das 'Gedächtnis' des Spectrum eingedrungen sind, daß dies die normale Art bei einem Computer ist, 2 Byte große Zahlen zu speichern; und in der Tat sind es die Zeilennummern, die seltsam abgespeichert werden.

ANDERE ZAHLEN

Alle anderen Zahlen werden im Programmbereich **zweimal** gespeichert. Zuerst werden Sie einfach als Zeichen gespeichert, (d.h. 1.2 wird als CHR$ 49;CHR$ 46;CHR$ 50 gespeichert). Dann wird ein CHR$ 14 dahintergespeichert, um anzuzeigen, daß diese Zeichen tatsächlich eine Zahl bilden, und dann wird der Wert der Zahl in den nächsten fünf Bytes gespeichert, indem entweder Ganzzahl- oder Fließkommaformat verwendet wird. Wir werden den Unterschied zwischen diesen beiden Formaten im Programm "VARIABLE" sehen, aber für den Augenblick genügt es zu sagen, daß, wo auch immer das Zeichen CHR$ 14 im Programmbereich auftaucht, wir und der Computer wissen, daß die nächsten fünf Bytes eine Zahl repräsentieren. (Siehe Zeile 50).

```
   10 LET p=PEEK 23635+256*PEEK 2
3636
   15 LET v=PEEK 23627+256*PEEK 2
3628
   20 LET n=256*PEEK p+PEEK (p+1)
   25 GO SUB 100: PRINT n;TAB 21;
"( ZEILEN": LET p=p+1: GO SUB 10
0: PRINT TAB 21;"( NUMMER": LET
p=p+1
   30 LET le=PEEK p+256*PEEK (p+1
)
```

```
00:  PRINT TAB 21;"( LAENGE": LET
 P=P+1
   40 LET le=p+le
   45 LET c=PEEK p
   50 GO SUB 100: IF c=14 THEN RE
STORE : PRINT TAB 21;"ZAHL :": F
OR j=1 TO 5: LET p=p+1: GO SUB 1
00: READ n$: PRINT TAB 21;n$: NE
XT j: GO TO 65
   55 IF c<16 OR c>23 THEN PRINT
TAB 16;CHR$ c: GO TO 65
   60 PRINT TAB 16;"(undruckbar)"
   65 LET p=p+1: IF p=v THEN STOP

   70 IF p=le THEN GO TO 20
   75 GO TO 45
  100 PRINT p;TAB 8;PEEK p;TAB 16
;: RETURN
  200 DATA "FUENF","BYTES","WERT"
,"DER","ZAHL"
9000 SAVE "READER 2" LINE 0
```

TRACE

Es ist sehr interessant, wenn man den Programmbereich des Spectrum 'lesen' kann, aber wozu nützt es? Bieten die beiden letzten Programme zusätzliche Möglichkeiten, die nicht durch den BASIC-Befehl LIST geboten werden?

Vielleicht nicht so, wie sie da stehen, aber wir kennen den genauen Speicherplatz von jedem einzelnen Programmteil und können den Platz eines jeden Befehls, einer Zahl oder Variablen ansprechen.

Wie hier aufgelistet, lokalisiert "TRACE" bestimmte Variablen, die in einem Programm verwendet wurden, und stellt Zeilennummer und Befehl fest, in denen sie vorkommt. Sie können das Programm leicht so ändern, daß es Befehlswörter oder Zahlen sucht, aber ich habe Variablen gewählt, weil Ihnen das vermutlich am meisten nützen wird.

(Einer der am häufigsten vorkommenden Programmfehler ist die zwei- oder mehrfache Verwendung des gleichen Variablennamens für verschiedene Variablen). Außerdem ist das Aufspüren von Variablen schwieriger als das Aufspüren von etwas anderem, da noch zwischen Variablen mit ähnlichen Namen (wie z.B. b, b$, bc oder ab) unterschieden werden muß. (Siehe Zeilen 75 und 80).

Einige andere Fallstricke, die es zu vermeiden gilt, wenn man eine bestimmte Variable aufspüren will, sind:
1) Bytes, die Teil einer Zeilennummer, Zeilenlänge oder eines numerischen Wertes sind, sollten ignoriert werden. Sie haben in "READER 1" gesehen, daß das Zeichen, welches durch diese Bytes dargestellt wird, irrelevant ist, wenn es aus dem Zusammenhang gerissen wird, und es könnte leicht das Zeichen sein, nach dem wir suchen (siehe Zeilen 35 und 55).

2) Alles, was zwischen Anführungszeichen steht, sollte ebenfalls ignoriert werden. Sie könnten ja ohne weiteres Wörter wie "a" in PRINT- oder INPUT-Befehlen verwenden wollen, ohne daß diese als Variable angesehen werden sollen. (Sie haben vielleicht auch schon herausgefunden, daß man mit einer kleinen Tastaturmanipulation Befehlswörter wie THEN oder RUN in Textzeilen einfügen kann. Diese werden durch den Computer zwar als ein Byte große Schlüsselwörter abgespeichert, aber Sie möchten natürlich nicht, daß sie durch ein "TRACE-Programm" als Befehls- oder Schlüsselwörter identifiziert werden.) Dieses Problem wird durch die Benutzung eines 'Anführungszeichenschalters' gelöst, q in den Zeilen 35, 45 und 50. q wird auf 1 gesetzt, wenn die Anführungszeichen 'aufgemacht' werden, und auf 0 wenn sie wieder 'zugemacht' werden.

Zeile 30 ist nicht unbedingt nötig, sie zeigt nur die gerade gelesene Zeile an, um Ihnen zu zeigen, daß das Programm, seitdem es zum letzten Mal 'fündig' wurde, nicht abgestürzt ist.

Außer nach der von Ihnen angegebenen Variablen, sucht das Programm auch nach dem Doppelpunkt, natürlich wieder nur außerhalb der Anführungszeichen, damit die Nummer des Befehls (c) innerhalb einer Zeile festgestellt werden kann.

```
    1 REM "TRACE"
    5 INPUT "Variable zum verfolg
en ";v$: LET l=LEN v$
   10 LET p=PEEK 23635+256*PEEK 2
3636
   15 LET v=PEEK 23627+256*PEEK 2
3628
   20 LET n=256*PEEK p+PEEK (p+1)
   25 LET le=PEEK (p+2)+256*PEEK
(p+3)
   30 PRINT n
   35 LET q=0: LET c=1: FOR j=4 T
O le+4
   40 LET p1=PEEK (p+j)
   45 IF p1=34 THEN LET q=NOT q
   55 IF p1=14 THEN LET j=j+5: GO
 TO 90
   60 IF p1=58 THEN LET c=c+1
```

```
0 90
  70 FOR k=1 TO l
  75 LET p1=PEEK (p+j+k-1): IF p
1<>CODE v$(k) THEN GO TO 90
  80 NEXT k: LET k=PEEK (p+j+k-1
): IF k=36 OR (k>47 AND k<58) OR
 (k>64 AND k<91) OR (k>96 AND k<
123) THEN GO TO 90
  85 PRINT FLASH 1; INK 2; PAPER
 7;n;: PRINT INK 9;TAB 5;":";c;T
AB 10;v$
  90 NEXT j
  95 LET p=4+p+le: IF p=v THEN S
TOP
 100 GO TO 20
9000 SAVE "TRACE" LINE 0
```

RENUMBER

Bis hierher haben wir in diesem Kapitel den RAM des Spectrum 'durchgelesen' und haben die genauen Speicherplatzadressen von bestimmten Bausteinen eines Basicprogrammes angesprochen.

Der nächste Schritt besteht darin, die Inhalte von einigen dieser Speicherplätze zu **verändern**. Statt nur mit PEEK den Programmbereich zu lesen, wollen wir jetzt mit POKE neue Werte hineinschreiben. Ein Basicprogramm, das sich verändert, während es abläuft! Vielleicht ist das einleuchtendste Beispiel dafür ein Zeilen-Renumerierungsprogramm.

Wir haben bereits gesehen, wie man Zeilennummern im Programmbereich erkennen kann. Sie bestehen **IMMER** aus zwei Bytes und werden **IMMER** von zwei weiteren Bytes gefolgt, die auf die nächste Zeilennummer deuten, oder bei der letzten Programmzeile auf VARS. Also, alles, was man tun muß, um ein Programm neu zu numerieren, ist, entsprechende neue Werte in die Speicherplätze, die die Originalnummern enthalten, hineinzuPOKEn. (Nicht vergessen, die neuen Nummern in zwei Bytes große Hexadezimalzahlen umzuwandeln – siehe Zeile 8).

Achtung: Diese Methode der Neunumerierung von Programmzeilen verändert die Ansprungadressen von GO TO und GO SUB **nicht**. Sie müssen diese manuell ändern, damit sie zu den neuen Zeilennummern passen. Aus diesem Grund läuft das Programm nicht mit sich selbst, und dieses wegen den Bedingungen in den Zeilen 1 und 3. Die erste, neu zu numerierende Zeile und ihre neue Nummer müssen hinter der letzten Zeile der 'Renumber-Routine' liegen. Es besteht keine Einschränkung bezüglich der letzten zu numerierenden Zeile, da das Programm entweder anhält, wenn es eine Zeilennummer erreicht, die hinter der als letzte Zeile angegebenen liegt, oder wenn es VARS

numerieren wollen, dann geben Sie auf die Anfrage "Letzte Zeile zum Renumerieren" '9999' ein.

```
    1 INPUT "Erste Zeile zum Renu
mmerieren ";f: IF f<10 THEN GO T
O 1
    2 INPUT "Letzte Zeile zum Ren
ummerieren ";e
    3 INPUT ("Neuer Wert fuer ";f
;" ");r: IF r<10 THEN GO TO 3
    4 INPUT "Schrittweite ";s: IF
 s<1 THEN GO TO 4
    5 LET p=PEEK 23635+256*PEEK 2
3636: LET v=PEEK 23637+256*PEEK
23626
    6 LET n=256*PEEK p+PEEK (p+1)
: LET l=PEEK (p+2)+256*PEEK (p+3
): IF n<f THEN GO TO 9
    7 IF n>e OR p=v THEN LIST : S
TOP
    8 LET p1=INT (r/256): POKE p,
p1: POKE p+1,r-(p1*256): LET r=r
+s
    9 LET p=4+p+l: GO TO 6
9000 SAVE "RENUMBER" LINE 0
```

GRÖSSEN-WECHSEL

Selbstverständlich kann man im Programmbereich mit POKE nicht nur Zeilennummern ändern. Jeder beliebige Programmbestandteil kann lokalisiert werden, und wenn er gefunden wird, kann man ihn auch ändern.

Es ist Ihnen überlassen, sich einige außergewöhnliche Anwendungen dafür auszudenken, hier jedenfalls ist ein nützliches Programm, das es gestattet, Programme entweder in großen oder in kleinen Buchstaben aufzulisten.

Sie werden schon bemerkt haben, daß alle bisherigen Programme in kleinen Buchstaben aufgelistet waren, um es Ihnen zu ermöglichen, zwischen Variablennamen und Befehlswörtern (die IMMER groß geschrieben sind), zu unterscheiden. Wenn Sie jedoch mit anderen Rechnern gearbeitet haben, besonders mit solchen, die keine Kleinschreibung ermöglichen, oder wenn Sie finden, daß Kleinbuchstaben wie "m" oder "l" schwer zu lesen sind, könnten Sie es vorziehen, Programmlistings mit "GRÖSSEN-WECHSEL" auf Großbuchstaben zu verändern, nachdem sie eingegeben sind. (Oder umgekehrt, sie mit CAPS SHIFT eingeben und sie dann entsprechend der Vorlage im Buch umzuwandeln).

"GRÖSSENWECHSEL" wandelt ALLE kleinen Buchstaben in große um (oder umgekehrt), einschließlich der Variablennamen und Text. Wenn Sie wollen, können Sie einen Schalter setzen (wie in TRACE), um Textpassagen unverändert zu lassen.

Die beiden Schlüsselzeilen in "Größenwechsel" sind:

Zeile 85: POKEd einen Großbuchstaben an die Stelle

Zeile 90: POKEd einen kleinen Buchstaben an die Stelle eines großen.

```
    1 REM groessenwechsel
    5 CLS : PRINT '''zum auflisten
n in kleinbuchstabenbitte ""l"",
  "
   10 PRINT '''zum auflisten in g
rossbuchstabenbitte ""u"" drueck
en"
   15 LET i$=INKEY$
   20 IF i$=CHR$ 108 OR i$=CHR$ 7
6 THEN LET l=1: GO TO 35
   25 IF i$=CHR$ 117 OR i$=CHR$ 8
5 THEN LET l=0: GO TO 35
   30 GO TO 15
   35 CLS : PRINT FLASH 1;"das pr
ogramm wird verarbeitet"
   40 DEF FN p(a)=PEEK a+256*PEEK
  (a+1)
   45 LET prog=FN p(23635)
   50 LET a=prog-5
   55 LET a=a+5
   60 LET p=PEEK a
   65 IF a>=FN p(23627) THEN CLS
: LIST : STOP
   70 IF p=13 THEN GO TO 55
   75 IF p=14 THEN LET a=a+1: GO
TO 55
   80 IF l THEN GO TO 90
   85 IF p>=97 AND p<=122 THEN PO
KE a,p-32: GO TO 95
   90 IF p>=65 AND p<=90 THEN POK
E a,p+32
   95 LET a=a+1: GO TO 60
9000 SAVE "wechsel" LINE 0

    1 REM GROESSENWECHSEL
    5 CLS : PRINT '''ZUM AUFLISTE
N IN KLEINBUCHSTABENBITTE ""L"",
  "
   10 PRINT '''ZUM AUFLISTEN IN G
ROSSBUCHSTABENBITTE ""U"" DRUECK
EN"
   15 LET I$=INKEY$
   20 IF I$=CHR$ 108 OR I$=CHR$ 7
6 THEN LET L=1: GO TO 35
   25 IF I$=CHR$ 117 OR I$=CHR$ 8
5 THEN LET L=0: GO TO 35
   30 GO TO 15
   35 CLS : PRINT FLASH 1;"DAS PR
OGRAMM WIRD VERARBEITET"
   40 DEF FN P(A)=PEEK A+256*PEEK
  (A+1)
   45 LET PROG=FN P(23635)
   50 LET A=PROG-5
```

```
65 IF A>=FN P(23627) THEN CLS
: LIST : STOP
  70 IF P=13 THEN GO TO 55
  75 IF P=14 THEN LET A=A+1: GO
TO 55
  80 IF L THEN GO TO 90
  85 IF P>=97 AND P<=122 THEN PO
KE A,P-32: GO TO 95
  90 IF P>=65 AND P<=90 THEN POK
E A,P+32
  95 LET A=A+1: GO TO 60
9000 SAVE "WECHSEL" LINE 0
```

VARIABLE

Inzwischen haben Sie wahrscheinlich genug vom Programmbereich, deshalb wenden wir uns dem Bereich zwischen VARS und E-LINE zu, der als Variablenbereich oder auch Variablenspeicher bekannt ist.

Sie werden auf den ersten Blick gesehen haben, daß dieses Programm sehr viel länger ist als alle vorhergehenden in diesem Kapitel. Der Grund liegt darin, daß Variablen in einem sehr verkürzten Format abgespeichert werden, und deshalb ist die 'Entschlüsselung' der Bytes im Variablenspeicher wesentlich komplizierter als die der Bytes im Programmbereich.

Das Byte (bzw. die Bytes), welche(s) den Variablennamen enthält, wird nicht einfach als der CODE des Namens gespeichert. Sie enthalten auch Informationen, um welche Art von Variable es sich handelt, und weiterhin die Anzahl und das Format der Bytes, die folgen. Die verschiedenen Arten von Variablen, die der Spectrum verarbeiten kann, sind unterhalb aufgezeigt (und es gibt mehr davon, als Sie glauben würden!)

ZAHLEN:

Ganzzahlige Variable mit einem Buchstaben als Namen
z.B. a
Ganzzahlige Variable mit zwei Buchstaben als Namen
z.B. ab
Ganzzahlige Variable mit mehrbuchstabigem Namen
z.B. abcd
Fließkommavariable mit einem Buchstaben als Namen
z.B. a
Fließkommavariable mit zwei Buchstaben als Namen
z.B. ab
Fließkommavariable mit mehrbuchstabigem Namen
z.B. abcd

z.B. a
Numerisches Array (Feldgruppe)
z.B. a()

ZEICHENKETTEN

Eindimensionale Zeichenkette
z.B. a$
Zeichenketten Array
z.B. a$()

(Beachten Sie, daß wir, die Benutzer, nicht durch einfaches Ansehen des Variablennamens unterscheiden können, ob es sich um eine ganzzahlige-, eine Fließkomma- oder um eine Schleifen-Kontrollvariable handelt. Wir können es nicht, aber der Spectrum kann es!)

Um genau zu erkennen, wie diese verschiedenen Arten von Variablen gesichert werden, empfiehlt es sich, das Folgende in Verbindung mit Kapitel 24 des Spectrum-Handbuches zu lesen (Seiten 160–170). Das Programm ist so geschrieben, daß die einzelnen Schritte, die zum Bestimmen des Typs der Variablen und ihres Wertes, genau in der gleichen Reihenfolge ablaufen, wie der Text des Handbuches.

Etwas ist allen Variablen gemeinsam, nämlich, daß sie mit einem Buchstaben beginnen müssen, und daß nicht zwischen großen und kleinen Buchstaben unterschieden wird. Das erlaubt nur 26 Möglichkeiten (mindestens für den ersten Buchstaben eines Namens), und jede Zahl bis 26 kann in nur fünf BITS gespeichert werden. (Wandeln Sie 26 in binäre Form um, und Sie werden es sehen!). Aus diesem Grund stehen noch drei Bits zur Verfügung, um den Variablentyp anzugeben.

Sie werden nicht daran gewöhnt sein, sich ein Byte als eine binäre, 8 Bit große Zahl vorzustellen, aber Sie müssen es, wenn Sie den Variablenspeicher verstehen wollen. Zeile 30 entfernt die 5 niederwertigsten Bytes vom Namensbyte (den Namen selbst) und hinterläßt eine Zahl im Wert

zwischen 2 und 7 (dezimal). Die REM-Anweisungen in den Zeilen 35 bis 60 zeigen an, welcher Typ von Variablen dadurch identifiziert worden ist, und außerdem enthalten diese Zeilen auch einen Sprungbefehl zu der entsprechenden Auswertungsroutine.

ROUTINE 100: ZAHLEN – NAME MIT EINEM BUCHSTABEN

Diese Routine ruft nur das Unterprogramm in Zeile 800 auf, das zunächst prüft, ob es sich um eine Ganzzahl oder um eine Fließkommazahl handelt. Die fünf Bytes, die auf den Namen folgen, enthalten den Wert der Variablen und sind genauso gespeichert wie im Programmbereich. Wenn das erste Byte den Wert null hat, dann ist die Zahl ganzzahlig, und die Auswertung ist relativ einfach (Zeile 825)! Die Zeilen 850 bis 870 enthalten die komplizierteren Berechnungen, die zum Entschlüsseln von Fließkommazahlen nötig sind.

ROUTINE 200: ZAHLEN – NAME AUS MEHREREN BUCHSTABEN

Zeile 200 enthält die Korrekturen, die nötig sind, um den ersten Buchstaben des Namens anzuzeigen. (Ich habe alle numerischen Variablen in kleinen Buchstaben und alle Zeichenketten in großen Buchstaben ausgeben lassen. Inzwischen sollten Sie erkennen, daß das rein willkürlich ist, und warum der Spectrum nicht zwischen den beiden unterscheidet.)

Zeile 230 identifiziert den letzten Buchstaben des Namens (höchstes Bit = 1). Zeile 240 gibt die mittleren Buchstaben aus. Zeile 260 druckt den letzten Buchstaben. Die Routine verzweigt dann zu Zeile 100 und wertet aus wie zuvor.

ROUTINE 300: NUMERISCHE ARRAYS (FELDGRUPPEN)

Auch Zeile 300 enthält die Korrekturen, die zum Anzeigen

900, das auch für Zeichenketten-Arrays verwendet wird, erkennt die Anzahl der Dimensionen (d), die Größe der einzelnen Dimensionen (el) und berechnet die Gesamtzahl der Elemente (t).

Das Unterprogramm 800 wird dann t mal aufgerufen, um jedes Element einzeln zu entschlüsseln.

ROUTINE 400: SCHLEIFENKONTROLLE

Der Variablenspeicher enthält weit mehr Informationen über Schleifen-Kontrollvariablen als nur ihren gegenwärtigen Wert.

Zeile 410 entschlüsselt die Variable wie üblich. In den Zeilen 425 bis 430 wird das Limit, in 435 bis 440 die Schrittweite und in den Zeilen 450 bis 460 die Zeilen- und Befehlsnummer der 'Schleifenzeile' gelesen.

ROUTINE 500: ZEICHENKETTEN

In Zeile 510 wird die Länge (LEN) der Zeichenkette bestimmt und die Zeilen 520 bis 540 geben die einzelnen Zeichen aus.

ROUTINE 600: ZEICHENKETTEN-ARRAY

Benutzt das Unterprogramm 900, um die Größe des Arrays auszugeben, und gibt dann das Zeichen aus jedem Element aus.

Das Programm ist sehr lang, aber wenn Sie es verwenden, werden Sie ziemlich viel darüber erfahren, auf welche Weise Daten im Computer gespeichert werden, und wie der Computer diese Daten interpretiert.

Das Programm läuft allein ab, obgleich die Ergebnisse nicht immer so sein werden, wie Sie es erwarten. (Denken Sie daran, daß der Inhalt des Variablenspeichers verändert wird, wenn das Programm läuft). Versuchen Sie, einige Scheinvariablen in Zeile 1 einzufügen, und untersuchen Sie,

wie diese gespeichert werden. Was passiert, wenn Sie folgendes eingeben:

1 LET x=1.1234567890123 45

```
    1 REM VARIABLEN
    5 DIM b(5)
   10 LET p=PEEK 23627+256*PEEK 2
3628
   20 PRINT p;TAB 6;: LET p1=PEEK
 p
   25 IF p1=128 THEN PRINT TAB 1;
"ENDE DES VARIABLENBEREICHS": ST
OP
   30 LET p2=INT (p1/32)
   35 IF p2=3 THEN GO TO 100: REM
 Zahl (1 Buchstabiger Name)
   40 IF p2=5 THEN GO TO 200: REM
 Zahl (Mehrbuchstabiger Name)
   45 IF p2=4 THEN GO TO 300: REM
 Numerisches Array
   50 IF p2=7 THEN GO TO 400: REM
 Schleifen Kontrolle
   55 IF p2=2 THEN GO TO 500: REM
 Zeichenkette
   60 IF p2=6 THEN GO TO 600: REM
 Alfanumerisches Array
   70 STOP : REM Fehler, wenn die
se Zeile erreicht wird
  100 REM Zahl:-
  105 PRINT CHR$ p1
  110 GO SUB 800
  115 PRINT TAB 16;n
  120 LET p=p+1: GO TO 20
  200 PRINT CHR$ (p1-64);: REM er
ster Buchstabe
  210 LET p=p+1
  220 LET p1=PEEK p
  230 IF p1>=128 THEN GO TO 260
  240 PRINT CHR$ p1;: REM mittler
er Buchstabe (falls vorhanden)
  250 GO TO 210
  260 PRINT CHR$ (p1-128);: REM l
etzter Buchstabe
  270 GO TO 110
  300 PRINT CHR$ (p1-32);"(";
  305 GO SUB 900
  310 FOR k=1 TO t
  315 GO SUB 800
  320 PRINT TAB 3;"Element #";k;T
AB 16;n
  325 NEXT k
  330 LET p=p+1: GO TO 20
  400 PRINT CHR$ (p1-128);
  410 GO SUB 800
  420 PRINT TAB 9;"(Schleife)";TA
```

```
 430 PRINT TAB 16;n;TAB 22;"Limi
t (TO)"
 435 GO SUB 800
 440 PRINT TAB 16;n;TAB 22;"STEP
"
 450 LET l=PEEK (p+1)+256*PEEK (
p+2): LET s=PEEK (p+3): LET p=p+
4
 455 PRINT TAB 16;l;TAB 22;"Zeil
e Nr."
 460 PRINT TAB 16;s;TAB 22;"Befe
hl"
 465 GO TO 20
 500 PRINT CHR$ p1;"$";TAB 16;
 510 LET l=PEEK (p+1)+256*PEEK (
p+2): LET p=p+2
 520 FOR j=1 TO l
 530 PRINT CHR$ PEEK (p+j);
 540 NEXT j: PRINT
 550 LET p=p+j: GO TO 20
 600 PRINT CHR$ (p1-128);"$(";
 605 GO SUB 900
 610 FOR k=1 TO t
 620 LET p=p+1
 630 PRINT TAB 3;"Element #";k;T
AB 16;CHR$ PEEK p
 640 NEXT k
 650 LET p=p+1: GO TO 20
 800 REM Ganzzahl
 805 FOR j=1 TO 5
 810 LET p=p+1: LET b(j)=PEEK p
 815 NEXT j
 820 IF b(1) THEN GO TO 845
 825 LET n=b(3)+256*b(4): IF b(2
)=255 THEN LET n=n-65536
 830 RETURN
 845 REM Fliesskomma
 850 LET m=1: IF b(2)<128 THEN L
ET m=0: LET b(2)=b(2)+128
 855 LET n=0: FOR j=2 TO 5: LET
n=n+b(j)/256↑(j-1): NEXT j
 860 LET b(1)=b(1)-128
 865 LET n=n*2↑b(1)
 870 IF m THEN LET n=-n
 875 RETURN
 900 REM Arrays
 905 LET p1=PEEK (p+1)+256*PEEK
(p+2): LET p=p+3
 910 LET d=PEEK p: LET p=p+1
 915 LET t=1: FOR j=1 TO d
 920 LET el=PEEK p+256*PEEK (p+1
): LET t=t*el: PRINT el;
 925 IF d<>1 AND j<>d THEN PRINT
 ",";
 930 LET p=p+2: NEXT j: PRINT ")
"
 940 LET p=p-1
```

DAS HERZ DER MASCHINE

RENUMBER
FARBE AUF ANFANGSWERT SETZEN
BLÄTTERN
SCROLL

RENUMBER

Alle Programme im vorangegangenen Teil laufen sehr langsam. Versuchen Sie ein sehr langes Programm mit dem Basic-Renumberprogramm neu zu numerieren, dann dauert es ewig - oder es kommt Ihnen wenigstens so vor.

Das hört sich nach einem Job für Maschinensprache an!

Hier ist das gleiche Programm, ganz leicht vereinfacht. (Diese Version renumeriert das gesamte Programm in einer Schrittweite von fünf, wobei die erste Zeile die Nummer 5 erhält. Sie können die Schrittweite durch Ändern des entsprechenden Bytes im Code, d.h. des 17. Zeichens im DATA-Befehl, verändern).

Das Basic-Ladeprogramm speichert das Maschinenprogramm ab der Adresse 32000 ab, es ist daher mit

RANDOMIZE USR 32000 aufzurufen.

Wenn Sie das Ladeprogramm mit GO TO 9000 abspeichern, werden Sie aufgefordert, sowohl das Ladeprogramm für Ihre Programmbibliothek (denn Sie werden es bestimmt nie mehr benötigen) als auch das Maschinensprache-Programm abzuspeichern.

Um das Maschinenprogramm zu einem späteren Zeitpunkt wieder zu laden, benutzen Sie:

CLEAR 31999:LOAD "RENUMBER"CODE 32000, 31

(Besitzer eines 48K Spectrum werden dieses Programm auf eine wesentlich höhere Adresse laden. Denken Sie dabei daran, daß man Bytes von einer Adresse SAVEn, und sie an eine andere Adresse laden kann. Es besteht also kein Grund dazu, das abgedruckte Ladeprogramm zu verändern.)

RENUMBER

Mnemonischer Code	Dezimal Code	Wirkung
LD HL, (23635)	42,83,92	HL = PROG
LD DE,0	17,0,0	DE = 0
LD BD, (23627)	237,75,75,92	BC = VARS
AND A	167	Übertragungs-register löschen
SBC HL,BC	237,66	Ist HL = VARS?
RET Z	200	Wenn ja, RETURN
ADD HL,BC	9	Wert von HL auf Anfangswert setzen
LD B,5	6,5	B = 5 (Schritt-weite)
INC DE	19	Erhöhen DE um B
DJNZ,-3	16,253	DE = neue Zeilen-nummer
LD (HL),D	114	Einfügen
INC HL	35	neue
LD (HL),E	115	Zeilennummer
INC HL	35	BC
LD C,(HL)	78	=
INC HL	35	Zeilen-
LD B,(HL)	70	länge
ADD HL,BC	9	HL = nächste
INC HL	35	Zeilennummer
JR,-25	24,231	Wiederholung

```
    1 RESTORE : CLEAR 31999: LET
j=32000
    2 READ a: IF a=9999 THEN STOP

    3 POKE j,a: LET j=j+1: GO TO
2
  500 DATA 42,83,92,17,0,0,237,75
,75,92,167,237,66,200,9,6,5,19,1
6,253,114,35,115,35,78,35,70,9,3
5,24,231
9000 SAVE "RENUMBER1" LINE 0
9005 SAVE "RENUMBER"CODE 32000,3
1
```

FARBE AUF ANFANGS-WERT SETZEN

Wie oft haben Sie schon ein Programm aufgeLISTet, nur um festzustellen, daß es in himmelblauer Schrift (INK) auf scharlachrotem Hintergrund (PAPER) ausgegeben wurde, oder noch schlechter, in schwarzer Schrift auf schwarzem Hintergrund?

Hier ist eine einfache Maschinenroutine, die die Farben des Bildschirms zurücksetzt auf den guten alten weißen Rand (BORDER), weißen Hintergrund (PAPER) und schwarze Schrift (INK), unabhängig davon, welche Farben durch das Programm hinterlassen worden waren.

Es ist nicht nur für die Fehlersuche sehr nützlich, es wäre auch sehr nett, wenn Sie dieses kleine Unterprogramm am Ende Ihrer Programme einfügen würden, damit der Benutzer Ihre Listings ohne schwere Augenschäden entziffern kann.

Alles, was das Programm macht, ist, die beiden Systemvariablen BORDER und ATTR P auf den Wert 56 zu setzen (was weiß für den Rand und Hintergrund bedeutet, oder, wie Sie aus ATTR TAFEL wissen, schwarze Schrift auf weißem Hintergrund.)

Nebenbei bemerkt: Es gibt noch einen anderen Weg, um Ihre Listings leserlich zu machen, ohne auf Maschinencode

Geben Sie Zeilennummer 1 ein
Drücken Sie beide Shift-Tasten und Taste "7"
Drücken Sie beide Shift-Tasten, halten Sie CAPS SHIFT weiter gedrückt und gleichzeitig Taste "Ø"
Drücken Sie ENTER

Zeile 1 wird als Leerzeile in Ihrem Listing erscheinen, aber alle folgenden Zeilen werden in schwarzer Schrift auf weißem Hintergrund ausgegeben.

Mnemonischer Code	**Dezimal Code**	**Wirkung**
LD BC,23524	1,72,92	BC = BORDER
LD A,56	62,56	56 = weißer Rand
LD (BC),A	2	Rand auf weiß setzen
LD C,141	14,141	BC = 23693 = ATTR P
LD (BC),A	2	Hintergrund und Schrift setzen
RET	201	RETURN

```
   10 CLEAR 31999
   20 RESTORE : FOR j=32000 TO 32
009
   30 READ a: POKE j,a: NEXT j: S
TOP
  100 DATA 1,72,92,62,56,2,14,141
,2,201
9000 SAVE "RESET" LINE 0
9005 SAVE "RESET"CODE 32000,10
```

BLÄTTERN

SPEICHERN UND WIEDERAUFRUFEN (48K)

Einige Rechner haben eine Blätterfunktion, wobei mehrere 'Bildschirminhalte' abgespeichert und nach Belieben wieder aufgerufen werden können.

Genau das Gleiche macht auch der Spectrum, wenn Sie dieses kurze Maschinenprogramm einsetzen.

"Store" kopiert den Inhalt des Bildschirmspeichers in einen reservierten Bereich des Arbeitsspeichers (oberhalb von RAMTOP), und "Recall" kopiert ihn einfach wieder zurück. Sie werden feststellen, daß die beiden Programme fast identisch sind, und beide lassen die Fähigkeit und Geschwindigkeit des LDIR Befehls erkennen, mit dem ca. 7000 Bytes im Bruchteil einer Sekunde übertragen werden können!

Das Basicprogramm ist mehr als nur ein Ladeprogramm, es enthält auch noch zwei Unterprogramme, die effektiv das Maschinenprogramm ändern, damit es den richtigen Teil des reservierten Speichers nach der benötigten 'Seite' absucht.

Das Unterprogramm 1000 speichert Bildschirmseiten ab den Adressen 33000, 40000, 47000 und 54000 (Seite 1-4).
Unterprogramm 2000 ruft jede dieser vier Seiten wieder auf.

Sie können "BLÄTTERN" in andere Programme einbauen, um Bildschirmausgaben nach Bedarf speichern und wieder aufrufen zu können, oder Sie können einen 'Katalog' Ihrer beliebtesten Bildschirmmasken aufbauen. Erinnern Sie sich daran, daß, wenn ein 'Bild' erst einmal in den reservierten

erzeugte, gelöscht werden kann, und Sie das Bild immer wieder mit GO SUB 2000 aufrufen können.

Zeile 9005 speichert nicht nur das Maschinenprogramm, sondern den **gesamten** Speicherbereich.

BLÄTTERN

SPEICHERN		WIEDERAUFRUFEN	
Mnemonischer Code	**Dezimal Code**	**Mnemonischer Code**	**Dezimal Code**
LD DE,33000	17,232,128	LD DE,16384	17,0,64
LD HL,16384	33,0,64	LD HL,33000	33,232,128
LD BC,6912	1,0,27	dto	dto
LDIR	237,176	dto	dto
RET	201	dto	dto

```
    1 REM Blaettern
   10 CLEAR 31999: LET s=32000: L
ET r=32050
   20 RESTORE : FOR j=0 TO 11: RE
AD a: POKE s+j,a: NEXT j
   30 FOR j=0 TO 11: READ a: POKE
 r+j,a: NEXT j: STOP
  100 DATA 17,232,128,33,0,64,1,0
,27,237,176,201
  200 DATA 17,0,64,33,232,128,1,0
,27,237,176,201
1000 INPUT "Speichern als Seite
(1-4) ";p
1010 LET p=INT p: IF p<1 OR p>4
THEN GO TO 1000
1020 LET a=p*7000+26000: LET a2=
INT (a/256): LET a1=a-a2*256
1030 POKE 32001,a1: POKE 32002,a
2
1040 LET n=USR s
1050 RETURN
2000 INPUT "Aufruf Seite No. (1-
4) ";p
2010 LET p=INT p: IF p<1 OR p>4
THEN GO TO 2000
2020 LET a=p*7000+26000: LET a2=
INT (a/256): LET a1=a-a2*256
2030 POKE 32054,a1: POKE 32055,a
2
2040 LET n=USR r
2050 RETURN
9000 SAVE "BLAETTERN" LINE 0
```

SCROLL

LINKS/RECHTS

Diese beiden Programme benutzen die Befehle RL (HL) und RR (HL), um den Bildschirm zu 'drehen', und erzeugen dadurch einen netten Scroll-Effekt nach rechts, bzw. nach links.

Das Basicprogramm enthält das Ladeprogramm (Zeilen 10–60) und außerdem eine Demonstration, wie das Programm arbeitet (Zeilen 5 und 100 bis 130).

Beachten Sie, daß der Endwert der FOR-TO-NEXT-Schleife (Zeilen 105 und 120) 8mal der Anzahl der Zeichen entspricht, um die der Schirm 'scrollt'.

Die Befehle RL und RR (Rotate Left / Rotate Right) haben einige "nahe Verwandte" in der Maschinensprache, und durch den Einsatz dieser können einige interessante Effekte erzielt werden (allerdings NICHT Scroll!). (Ändern Sie die **ACHTE** Data Position in den Zeilen 50 und 60).

Versuchen Sie:

RLC/RRC (Rotate and Carry)
Ändern Sie "22" in "6" in Zeile 50. Ändern Sie "30" in "14" in Zeile 60. (Erzeugt ein "Kräuseln" des Bildschirms, während sich jeder Buchstabe um seine Achse dreht.)

Shift (SLA/SRA)
Ändern Sie "22" in "38" in Zeile 50 **oder** "30" in "46" in Zeile 60. (Erzeugt eine Art Zugjalousie, während jeder einzelne Buchstabe vom Bildschirm herunterwandert).

SCROLL

LINKS		RECHTS	
Mnemonischer Code	**Dezimal Code**	**Mnemonischer Code**	**Dezimal Code**
LD HL,22527	33,255,87	LD HL,16384	33,0,64
LD C,32	14,32	dto	dto
AND A	167	dto	dto
RL (HL)	203,22	RR (HL)	203,30
DEC HL	43	INC HL	35
DEC C	13	dto	dto
JRNZ,-5	32,250	dto	dto
LD A,63	62,63	LD A,88	62,88
CP H	188	dto	dto
JRNZ,-13	32,242	dto	dto
RET	201	dto	dto
NOP	0		
NOP	0		

```
    5 BORDER 6: PAPER 6: INK 2: C
LS
   10 CLEAR 31999
   20 LET j=32000
   30 READ a: IF a=9999 THEN GO T
O 100
   40 POKE j,a: LET j=j+1: GO TO
30
   50 DATA 33,255,87,14,32,167,20
3,22,43,13,32,250,62,63,188,32,2
42,201,0,0
   60 DATA 33,0,64,14,32,167,203,
30,35,13,32,250,62,88,188,32,242
,201
  100 FOR j=1 TO 22: PRINT "qwert
yuiopasdfghjklzxcvbnm123456": NE
XT j
  105 FOR j=1 TO 8*16
  110 RANDOMIZE USR 32020
  115 NEXT j
  120 FOR j=1 TO 8*16
  125 RANDOMIZE USR 32000
  130 NEXT j: STOP
9000 SAVE "SCROLL" LINE 0
9005 SAVE "SCROLL"CODE 32000,38
9999 DATA 9999
```

INTERMEZZO

OVERS 1
OVERS 2
MUSTER 1
MUSTER 2
INVERT

OVERS 1

In diesem Buch gibt es eine Menge Beispiele für selbstdefinierte Grafikzeichen, aber wenn Sie nur ein komisch aussehendes Zeichen für irgend einen Zweck erzeugen wollen, dann gibt es eine einfache Möglichkeit das zu tun, ohne den ganzen CHR$ 144-Kram. Sie müssen nur die EXKLUSIV-ODER Möglichkeit des Ausdrucks OVER 1 benutzen, um ein Zeichen über ein anderes zu zeichnen, wobei alle Punkte, die beiden Zeichen gemeinsam sind, gelöscht werden.

Dieses Programm gestattet Ihnen, zwei beliebige Zeichen einzugeben, und Zeile 50 zeigt Ihnen die beiden Zeichen und das seltsame EXKLUSIV-ODER Zeichen, das durch OVER 1 erzeugt wurde.

Sie werden in der Tat überrascht sein, wie einfach es ist, mit dieser Methode einige interessante "Weltraummonster" zu gestalten.

```
   10 PRINT "EXPERIMENTE MIT "" O
VER """;
   20 INPUT "geben Sie 2 beliebig
e Zeichen    (oder SPACE fuer End
e)ein ";a$
   30 IF a$=CHR$ 32 THEN STOP
   40 IF LEN a$<>2 THEN GO TO 20
   50 PRINT a$(1);CHR$ 8; OVER 1;
a$(2);"  = ";a$(1);   "   OVER    ";
a$(2)
   60 GO TO 20
9000 SAVE "OVERS" LINE 0
```

OVERS 2

Das zweite Programm ist ein Spiel (ein Sortierspiel), entworfen, um Sie mit den Eigenschaften des Befehls OVER 1 vertrautzumachen.

Zeile 30 legt eine unvollständige Reihe von $-Zeichen über die Mitte des Bildschirms.

Sie können das blinkende $-Zeichen kontrollieren und es mit Hilfe der Pfeiltasten (5 und 8) nach links und rechts über die anderen hinwegbewegen.

Endziel des Spiels ist es, entweder eine vollständige Reihe von $-Zeichen, oder eine völlig leere Zeile zu erzeugen.

Als Zwischenziel jedoch könnten Sie versuchen zu erklären, was genau passiert, warum einige Zeichen gelöscht werden und andere neu erzeugt und welche Eingreifmöglichkeiten Sie haben, um diese Ereignisse zu veranlassen.

```
  10 OVER 1: BORDER 1: PAPER 5:
INK 9: CLS
  20 LET b=0
  30 FOR j=1 TO 31: PRINT AT 11,
j;"$": LET j=j+RND*3: NEXT j
  40 LET b=b+(INKEY$="8" AND b<>
31)-(INKEY$="5" AND b<>0)
  50 PRINT AT 11,b;"$"
  60 GO TO 40
9000 SAVE "OVERS 2" LINE 0
```

MUSTER 1

Ich bin sicher, daß Ihnen viel nützlichere Dinge einfallen, als hübsche Muster auf dem Bildschirm Ihres Spectrum zu erzeugen.

Hier sind zwei Programme, mit denen Sie genau das tun können.

Das erste Programm erzeugt Kaleidoskop-Muster. Zeile 60 zeichnet die vier Symmetriepunkte. OVER 1 (in Zeile 10) bedeutet, daß Punkte sowohl mit PLOT gezeichnet, als auch mit UNPLOT gelöscht werden. Der Bildschirm wird dadurch niemals ganz voll.

```
   10 OVER 1: RANDOMIZE
   20 BORDER 0: PAPER 0: CLS
   30 LET x=RND*128
   40 LET y=RND*88
   50 INK RND*7: BRIGHT INT (RND*
2)
   60 PLOT x,y: PLOT 255-x,y: PLO
T x,175-y: PLOT 255-x,175-y
   70 IF INKEY$="" THEN GO TO 30
   80 STOP
9000 SAVE "MUSTER" LINE 0
```

MUSTER 2

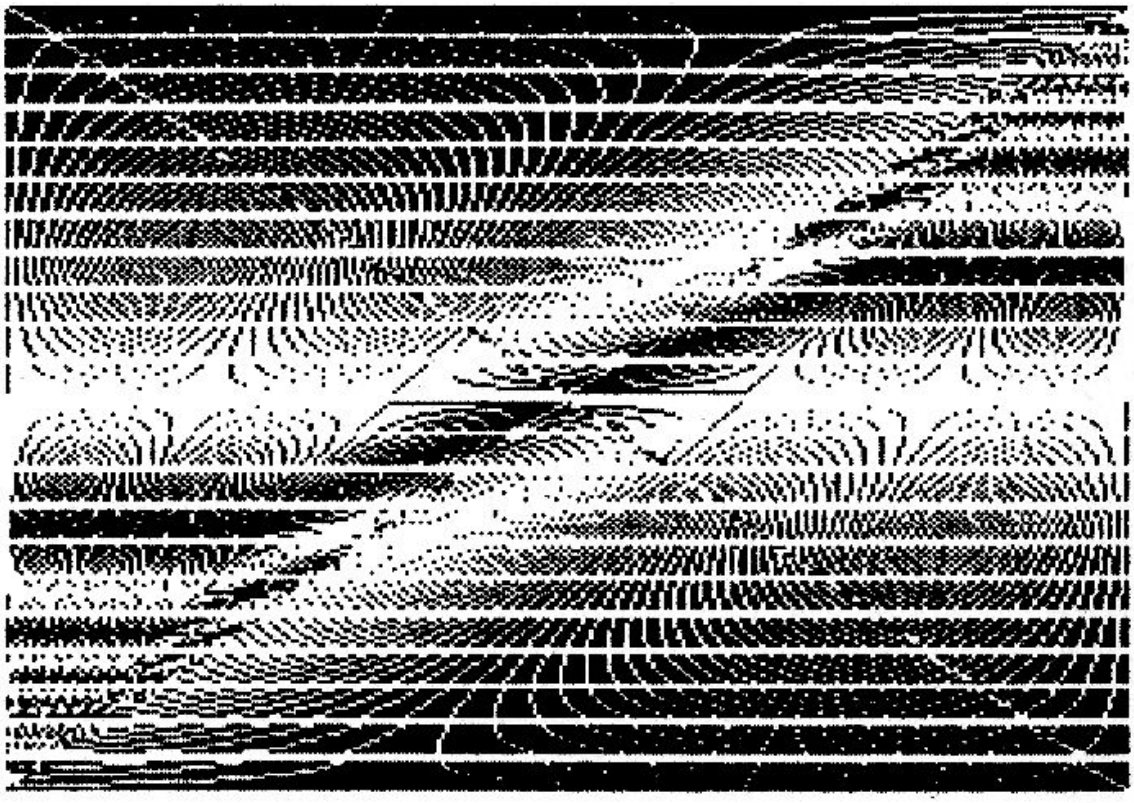
"Muster 2" erzeugt geflammte oder "Interferenzmuster" die durch das Zeichnen von fast paralellen Linien unter Verwendung von OVER 1 entstehen.

Merke:

Es ist schwierig, ein 'Interferenzmuster-Programm' zu schreiben, das kontinuierlich läuft, da der OVER 1-Befehl gewöhnlich die Muster vom Bildschirm löscht.

Wenn man jedoch nacheinander vier Muster durchläuft, dann kann der Schirm nie vollständig gelöscht werden.

Die Farbe der Schrift (INK) wird am Ende jedes Zyklus von 4 Mustern in Zeile 20 geändert. Befehl 2 in Zeile 20 verhindert, daß die gleiche Farbe (für INK) wiederholt werden kann.



```
    5 OVER 1: BORDER 0: PAPER 0:
INK 2: CLS
   10 GO SUB 100
   15 LET a=2
   20 LET i=INT (RND*7)+1: IF i=a
 THEN GO TO 20
   25 INK i: LET a=i: GO SUB 200:
 GO SUB 100: GO SUB 100: GO SUB
200: GO TO 20
  100 FOR x=0 TO 255
  105 PLOT x,0: DRAW 255-x,175: P
LOT 255-x,175: DRAW x-255,-175
  110 NEXT x
  115 RETURN
  200 FOR x=0 TO 255
  205 PLOT x,175: DRAW 255-x,-175
: PLOT 255-x,0: DRAW x-255,175
  210 NEXT x
  215 RETURN
9000 SAVE "MUSTER 2" LINE 0
```

INVERT

Dieses Programm gehört eigentlich nicht in dieses spielerische Kapitel, da es eine sehr nützliche Anwendungsmöglichkeit des Spectrum erläutert. Allerdings ist das Beispiel, welches ich ausgesucht habe, um diese Fähigkeit zu demonstrieren, unglaublich trivial.

Sie werden annehmen, daß man mit dem Spectrum nur 21 Graphikzeichen selbst definieren kann, aber man kann tatsächlich den gesamten Zeichenersatz selbst definieren. Wenn Sie also in russischer, griechischer oder sogar chinesischer Schrift drucken wollen, dann können Sie das auch tun.

Die Technik besteht darin, den Inhalt der Systemvariablen CHARS so zu verändern, daß nicht mehr wie normal eine Adresse im ROM, sondern eine andere Adresse (vorzugsweise oberhalb von RAMTOP) darin enthalten ist. Dort sind dann Ihre selbstdefinierten Zeichen gespeichert. (Siehe Zeile 70).

Dieser neue Zeichensatz wird auf die gleiche Weise aufgebaut wie alle vom Benutzer definierten Graphikzeichen, d.h., jeweils acht Bytes beschreiben ein Zeichen. Für dieses Programm habe ich allerdings geschwindelt, indem ich eine exakte Kopie des normalen Zeichensatzes erzeugt habe. Besser gesagt: FAST eine exakte Kopie, denn in Zeile 40 wird sichergestellt, daß alle Zeichen auf dem Kopf stehend kopiert werden.

Das Ergebnis ist ganz amüsant und könnte einige Bestürzung auslösen, wenn Sie das Programm – natürlich nur aus Versehen – auf dem Spectrum eines Freundes oder in einem Computerladen laufen lassen.

Die Tatsache, daß der Computer jetzt alles, einschließlich seiner Systemmeldungen, auf dem Kopf stehend ausgibt,

haupt nicht. Sie können sogar andere Programme laden und laufen lassen, die perfekt ablaufen, aber die umgedrehten Zeichen verwenden.
Das Programm benötigt mehrere Sekunden Laufzeit, aber wenn der neue Zeichensatz erst einmal erzeugt ist, können Sie ohne Verzug von einem Satz zum anderen umschalten. Geben Sie GO TO 9995 ein, um zu den normalen Zeichen zurückzukehren, und GO TO 70, um den umgedrehten Satz zu erhalten.

```
    5 CLEAR 29999
   10 LET c=PEEK 23606+256*PEEK 2
3607+256
   20 FOR j=0 TO 95
   30 FOR p=0 TO 7
   40 POKE 30000+7-p+8*j,PEEK (c+
p+8*j)
   50 NEXT p
   60 NEXT j
   70 POKE 23606,46: POKE 23607,1
16
   80 STOP
9000 SAVE "INVERT" LINE 0
9995 REM zurueckschalten
9996 POKE 23606,0: POKE 23607,60
```

AUFGABE

Können Sie diese 16 Punkte mit nur 6 geraden Linien verbinden, ohne den Bleistift abzusetzen? (Die Linien dürfen über das Quadrat hinausgehen!)

Die Lösung finden Sie auf Seite 141.

ROUTINEN

INPUT 1
INPUT 2
SORT
BUNDESLIGA

CASH
EINEN DATZ FINDEN

INPUT 1 + 2

Die Art, wie der Spectrum den INPUT-Befehl verarbeitet, ist in mancher Hinsicht schwächer als bei anderen Rechnern.

Der Text des INPUT-Befehls erscheint unten auf dem Schirm, und sobald Sie Enter drücken, verschwinden sowohl der Text als auch die Eingabe. Das ist nicht immer erwünscht, besonders in "Fragebogenprogrammen", wo der Benutzer vielleicht bei früheren Fragen und Antworten nachsehen möchte.

INPUT 1 und INPUT 2 sind zwei Unterprogramme, die beide je eine Alternative zu der üblichen INPUT-Routine bieten. INPUT 1 dupliziert die übliche Form des INPUT-Befehls, die bei den meisten Computern verwendet wird, wobei Anfrage und Antwort an einer bestimmten Stelle des Schirm erscheinen und dort auch bleiben, nachdem der INPUT-Vorgang abgeschlossen wurde.

INPUT 2, auch bekannt als 'Eingabe à la "Auf Los gehts los"', bringt keine Anfrage, sondern zeichnet eine Reihe von Gedankenstrichen, die nach Drücken der entsprechenden Tasten durch Buchstaben ersetzt werden. Die Eingabe ist entweder durch ENTER oder bei Erreichen des Endes der Gedankenstrich-Reihe beendet. Diese Methode ist ideal für die Eingabe von Daten in eine Zeichenketten-Feldgruppe (String array), wo die Länge der Elemente vorgegeben ist.

ACHTUNG: Die erste Zeile jedes Unterprogramms erstellt lediglich 'örtliche' Variablen zur Demonstration der Wirkungsweise. Wenn Sie die Routinen in anderen Programmen verwenden, müssen Sie die Zeilen 8000/8100 so verändern, daß:

- x und y Zeile und Spalte sind, wo die Eingabe erscheinen soll.
- in INPUT 1 t$ die Anfrage ist
- in INPUT 2 t$ die Zeichenketten-Feldgruppe ist, für die Daten eingegeben werden sollen.
- 1 die Länge der Zeichenketten-Feldgruppe ist.
- beide Unterprogramme die Eingabe in t$ ausgeben.

BEMERKUNGEN ZUM PROGRAMM

Beide Programme verwenden 'Verkettung', um die Zeichenkette t$ umzuwandeln. Jeder neue Wert von INKEY$ wird am Ende von t$ hinzugefügt, außer die gedrückte Taste war entweder ENTER (CHR$ 13), was die Eingabe beendet, oder DELETE (CHR$ 12), wodurch das letzte Zeichen gelöscht wird (Zeilen 8040 und 8130).

```
8000 LET x=10: LET y=0: LET t$=""
Name"
8005 PRINT AT x,y;t$
8010 LET y=y+2+LEN t$
8015 LET t$=""
8020 IF y>31 OR y=-1 THEN LET y=
31*(y=-1): LET x=x-1+2*(y=0)
8025 PRINT AT x,y; FLASH 1;"?"
8030 PAUSE 0: LET q$=INKEY$
8035 IF q$=CHR$ 13 AND t$<>"" TH
EN GO TO 8065
8040 IF q$=CHR$ 12 AND t$<>"" TH
EN PRINT AT x,y;CHR$ 32: LET y=y
-1: LET t$=t$( TO LEN t$-1): GO
TO 8020
8045 IF q$<CHR$ 32 THEN GO TO 80
30
8050 LET t$=t$+q$
8055 PRINT AT x,y;q$: LET y=y+1
8060 GO TO 8020
8065 PRINT AT x,y;CHR$ 32: RETUR
N
9000 SAVE "INPUT 1"
```

```
8100 LET x=5: LET n=10: LET l=15
: DIM t$(n,l)
8105 FOR j=1 TO n: FOR k=1 TO l
8110 PRINT AT x+j-1,k;CHR$ 45
8115 NEXT k: LET k=1
8120 PAUSE 0: LET q$=INKEY$
8125 IF q$=CHR$ 13 AND k<>1 THEN
 GO TO 8150
8130 IF q$=CHR$ 12 AND k<>1 THEN
 LET k=k-1: PRINT AT x+j-1,k;CHR
$ 45: LET t$(j)=t$(j, TO k): GO
TO 8120
8135 IF q$<CHR$ 32 THEN GO TO 81
20
8140 LET t$(j,k)=q$: PRINT AT x+
j-1,k;q$: LET k=k+1
8145 IF k<=l THEN GO TO 8120
8150 NEXT j
8155 RETURN
9000 SAVE "INPUT 2" LINE 0
```

SORT

Hier haben wir eine einfache Methode, um eine Feldgruppe mit Zahlen in absteigender Reihenfolge zu sortieren (oder, mit geringen Änderungen, in aufsteigender Reihenfolge).

Die Methode ist eine Variation des bekannten 'Bubble-Sort' und funktioniert folgendermaßen:

Es gibt zwei verschachtelte Schleifen, wobei die Äußere (j) die Anzahl der 'Durchgänge' durch die Feldgruppe festlegt.

Die innere Schleife (k) vergleicht jede Zahl in der Feldgruppe mit der folgenden Zahl, und wenn die zweite größer als die erste ist, vertauscht sie diese. Am Ende des ersten Durchgangs bleibt deshalb die kleinste Zahl als letztes Element der Feldgruppe übrig.

Beim nächsten Durchgang (NEXT j) sortiert die k-Schleife die zweitkleinste Zahl in das zweitletzte Feldgruppenelement usw. bis zum letzten Durchgang, in dem nur noch die beiden ersten Elemente sortiert werden.

Die Zeilen 8200 und 8218 gehören nicht zum Sortierprogramm, sondern dienen der Demonstration. Zwanzig zufällig ausgewählte Zahlen werden auf der linken Seite des Bildschirms ausgegeben, sortiert, und dann in absteigender Reihenfolge auf der rechten Bildschirmhälfte aufgelistet.

```
8200 LET n=20: DIM a(n): FOR j=1
 TO n: LET a(j)=INT (RND*1000):
PRINT j;TAB 5;a(j): NEXT j
8205 FOR j=1 TO n-1: FOR k=1 TO
n-j
8210 IF a(k)<a(k+1) THEN LET t=a
(k): LET a(k)=a(k+1): LET a(k+1)
=t
8215 NEXT k: NEXT j
8218 FOR j=1 TO n: PRINT AT j-1,
16;j;TAB 22;a(j): NEXT j: REM (N
ur zur Demonstration)
8220 RETURN
```

BUNDES-LIGATABELLE

"Liga" ist eigentlich kein Unterprogramm, sondern ein komplettes Programm, und es enthält eine modifizierte Abart des vorangegangenen Sortierprogramms. Diese kann in viele andere Programme eingefügt werden, wo eine regelmäßig aktualisierte 'Punktetabelle' verlangt wird, z.B. Fußball, Wahlergebnisse usw.

Im nebenstehenden Listing enthält die DATA-Zeile 8000 die Zahl 18, gefolgt von den Namen der 18 Mannschaften der ersten Bundesliga. Wofür Sie das Programm auch benutzen, die DATA-Zeile sollte immer aus einer Zahl, gefolgt durch diese Anzahl von Posten, bestehen.

Zeile 10 liest die Daten in das Array t$() ein, aber das erste Zeichen jedes Elements ist für die PUNKTE von jeder 'Mannschaft' reserviert.

Der Hauptteil des Programms enthält jedoch eine andere Form der Eingabe-Routine, bei der das Drücken jeder numerischen Taste als Eingabe gesehen wird, während ein Druck auf DELETE den Namen des letzten Teams und die Punkte des vorletzten löscht, so daß diese geändert werden können.

Das Sortierungsprogramm entspricht dem letzten Programm, außer daß hier das erste Zeichen jedes Elements herausgezogen und in eine Zahl umgewandelt wird (unter Verwendung von CODE). Die gesamte Feldgruppe wird dann nach den Werten dieser Zahlen sortiert.

Offensichtlich funktioniert diese Methode nur, wenn keine der Mannschaften mehr als 255 Punkte erzielen kann. Das

Anwendung ein höheres Ergebnis möglich sein sollte, muß ein Zweibyte-System verwendet werden.

Nach der Ausgabe der sortierten Daten wird das Programm einschließlich Daten gespeichert (Zeile 240), damit nach dem nächsten Laden neue Punkte zu den bisherigen Ergebnissen hinzuaddiert werden können.

```
    1 REM Bundesliga Tabelle
   10 RESTORE 8000: READ n: DIM t
$(n,20): FOR j=1 TO n: LET t$(j,
1)=CHR$ 0: READ t$(j,2 TO ): NEX
T j
   20 DIM b$(20)
  100 CLS : PRINT "Eingabe der Pu
nkte der Woche:"
  105 FOR j=1 TO n
  110 LET x=INT ((j+1)/2): LET y=
16*(j=x*2)
  115 PRINT AT x,y;t$(j,2 TO );CH
R$ 32;CHR$ 8;
  120 PAUSE 0: LET p$=INKEY$
  125 IF p$=CHR$ 12 AND j>1 THEN
PRINT AT x,y;b$: LET j=j-1: LET
t$(j,1)=CHR$ (CODE t$(j,1)-p): G
O TO 110
  130 IF p$<"0" OR p$>"9" THEN GO
 TO 120
  135 PRINT p$
  140 LET p=VAL p$: LET t$(j,1)=C
HR$ (CODE t$(j,1)+p)
  150 NEXT j: PRINT AT 15,11;"Sor
t laeuft"
  155 GO SUB 8200
  200 CLS : FOR j=1 TO n
  205 IF j<>1 THEN IF t$(j,1)=t$(
j-1,1) THEN GO TO 215
  210 PRINT j;
  215 PRINT TAB 3;t$(j,2 TO );COD
E t$(j,1)
  220 NEXT j
  230 INPUT "Druecken Sie ""ENTER
""um die     Daten zu speichern";
a$
  240 SAVE "LIGA" LINE 100
8000 DATA 18,"Hamburger SV","Bay
ern Muenchen","VfB Stuttgart","W
erder Bremen","Borussia Dortmund
","1.FC Koeln","1.FC Kaiserslaut
ern","1.FC Nuernberg","Eintr.Fra
nkfurt","Braunschweig","Fortuna
Duesseldorf","Arminia Bielefeld"
,"VfL Bochum","Bayer Leverkusen"
,"Bor.M'gladbach","Hertha BSC","
FC Schalke 04","Karlsruher SC"
```

```
8205 FOR j=1 TO n-1: FOR k=1 TO
n-j
8207 LET a=CODE t$(k,1): LET b=C
ODE t$(k+1,1)
8210 IF a<b THEN LET a$=t$(k): L
ET t$(k)=t$(k+1): LET t$(k+1)=a$
8215 NEXT k: NEXT j
8220 RETURN
9000 SAVE "LIGA" LINE 0
```

CASH

Als eines der nützlichsten Unterprogramme überhaupt wandelt "CASH" unformatierte Zahlen in ein festes 'Kassenformat' um, so daß sie vernünftig in Spalten untereinander geschrieben werden können.

Wenn das Unterprogramm aufgerufen wird, sollte die Variable p eine Zahl enthalten. Nach dem Rücksprung aus dem Unterprogramm steht in m$ die formatierte Zahl.

Die sechste Stelle von m$ ist immer ein Dezimalpunkt, deshalb werden alle Zahlen, auch Ganzzahlen mit 2 Dezimalstellen, ausgegeben. Das Programm enthält außerdem einen Fehlerschalter (Variable x). Wenn p nicht als gültige Zahl dargestellt werden kann (mehr als 2 Dezimalstellen), wird x auf 1 gesetzt, so daß das Hauptprogramm die Zahl zurückweisen kann.

Das Programm enthält keine Demonstrationsdaten, da es in zwei Programmen in diesem Buch enthalten ist "FAKTURA" und "RECHNUNGS-BUCH". Aber wenn Sie es testen wollen, dann fügen Sie einfach die Zeilen:

```
10 INPUT p: GO SUB 5000
20 if x=0 THEN PRINT m$
30 GO TO 10
```

ein.

```
    1 REM CASH
5000 LET c=0: DIM m$(8): LET m$=
STR$ p
5010 IF m$(8)=CHR$ 32 THEN LET m
$=CHR$ 32+m$( TO 7): GO TO 5010
5020 FOR j=1 TO 8
5030 IF m$(j)="." THEN GO TO 505
0
5040 NEXT j
5050 IF j=9 THEN LET m$=m$(4 TO
)+".00"
5060 IF j=7 THEN LET m$=m$(2 TO
)+".0"
```

EINEN SATZ FINDEN

Hierbei handelt es sich um ein wichtiges Unterprogramm für jedes Datenbankprogramm.

Zeile 8500, die übliche Zeile 'nur für Demonstrationszwecke', erlaubt es Ihnen, eine Mini-Datenbank, bestehend aus 10 Sätzen und 15 Feldern pro Satz, aufzubauen. Die tatsächliche Größe der Datenbank ist unwichtig, da alles, was von Ihnen in Zeile 8500 verlangt wird, die Eingabe des ersten Feldes (des Schlüsselfeldes) von jedem Satz ist.

Das Programm fordert Sie auf, den Namen des Satzes, den Sie verarbeiten wollen, einzugeben, und durchsucht dann sämtliche Schlüsselfelder, bis es diesen Namen gefunden hat. Nichts Besonderes dabei ... aber was das Programm so nützlich macht, ist die Tatsache, daß, wenn Sie nur den Teil eines Namens eingeben (also nur einen oder zwei Buchstaben oder auch nur ein Leerzeichen), es alle Schlüsselfelder sucht, die diese Zeichen enthalten.

Stellen Sie sich beispielsweise vor, Sie hätten eine Informationsdatei aufgebaut, die verschiedene Rechner betrifft. Ihre Schlüsselfelder würden aus Namen wie "Spectrum", "Commodore", "Apple" oder "Acorn" usw. bestehen.

Monate später, wenn Sie sich die Datei wieder einmal ansehen wollen, haben Sie vielleicht vergessen, ob die Informationen über den Spectrum unter "Spectrum" oder unter "Sinclair" enthalten ist. Das spielt keine Rolle, geben Sie einfach "S" ein, und das Programm wird sie finden.

Und wird "Commodore" mit einem oder mit zwei m geschrieben? Wer fragt danach? Einfach "Com" eingeben!

Aber was passiert, wenn Sie "A" eingeben? Findet das Programm dann "Apple" oder "Acorn"?

Es findet das was zuerst kommt, aber es bittet Sie um eine Bestätigung, daß es den richtigen Satz gefunden hat. Wenn es also "Apple" findet, und Sie wollten "Acorn", dann drücken Sie einfach "n", und es wird mit der Suche fortfahren.

Wenn es dem Programm nicht gelingt, einen passenden Satz zu Ihrer Eingabe zu finden, erscheint die Meldung 'Satz nicht gefunden', und Sie werden ersucht, eine neue Eingabe zu machen.

Ein Tip:

Ich empfehle Ihnen, alle Schlüsselfelder mit einem gemeinsamen Zeichen beginnen zu lassen, also einem Leerzeichen oder einem Punkt. Wenn Sie dieses Zeichen auf die Anfrage "Name des Satzes" eingeben, werden alle Sätze nacheinander angezeigt und Sie erhalten so eine Art Inhaltsverzeichnis für Ihre Datenbank.

```
8500 REM  einen Satz finden
8505 LET n=10: DIM a$(n,15,15):
FOR j=1 TO n: INPUT a$(j,1): NEX
T j
8510 INPUT "Name des Satzes";t$
8520 IF t$="" THEN GO TO 8510
8530 FOR j=1 TO n
8540 FOR k=1 TO 15-LEN t$
8550 IF a$(j,1,k TO k+LEN t$-1)=
t$ THEN GO TO 8600
8560 NEXT k: NEXT j
8570 PRINT FLASH 1;"SATZ NICHT G
EFUNDEN": GO TO 8510
8600 PRINT "GEFUNDEN: ";a$(j,1)
8610 INPUT "richtig (J/N) ?";q$
8620 IF q$="n" THEN GO TO 8560
8630 IF q$="j" THEN RETURN
8640 GO TO 8610
9000 SAVE "FINDEN" LINE 0
```

COMPUTER-KUNST

MARYLIN
MALEN NACH NUMMERN
STERNENBANNER
MELODIE 1
MELODIE 2

MARYLIN

Um dieses Programm einzutippen, braucht man eine ziemlich lange Zeit, aber es ist die Anstrengung wert!

Es ist eine wundervolle Demonstration der Sinclair Farbgraphik, keine Strichzeichnung oder Karikatur, sondern ein farbiges Bildschirmportrait.

Und ich meine SINCLAIR Graphik, denn das Bild ist ausschließlich aus den acht Standardgraphikzeichen aufgebaut, keine hochauflösende Graphikpunkte oder selbstdefinierte Zeichen.

Nur acht Graphikzeichen? Sicher! Zwar stellen CHR$ 128 bis CHR$ 143 alle Graphikzeichen dar, aber acht davon sind nur die Umkehrung der anderen. Da das Programm für jedes Zeichen Hinter- und Vordergrundfarben separat festlegt, hat das Prinzip der Umkehrung keine Bedeutung, und deshalb werden CHR$ 136 bis CHR$ 143 nicht benötigt. (Tatsächlich wird auch CHR 128 nicht gebraucht, da es das gleiche Zeichen wie CHR$ 32, also SPACE, darstellt – das Bild besteht also in Wirklichkeit nur aus SIEBEN Graphikzeichen plus Leerzeichen).

Die erste Stufe beim Gestalten des Programms war das Aufzeichnen des Portraits auf Zeichenpapier. Um es auf den Bildschirm zu befördern, waren zwei weitere Operationen nötig:

1) Aufbau einer Feldgruppe, die die benötigten Graphikzeichen enthält.

2) Hinzufügen der Farben durch einPOKEn in die Attributdatei.

DIE FELDGRUPPE

Auf dem Bildschirm gibt es 704 Zeichenpositionen, also ist a$ eine Feldgruppe mit 704 Elementen, von denen jedes einer bestimmten Bildschirmposition entspricht, d.h., wenn es gedruckt bzw. ausgegeben wird, dann ist a$ (32) die rechte obere Ecke des Bildschirms und a$ 352 ist irgendwo in der Mitte.

Es gibt keinen Grund, warum Sie a$ nicht mit je einem Zeichen für jedes Element laden sollten. Ihr Programm würde sehr lang werden, aber Sie könnten immer die 'Zwei-Programm-Technik' anwenden (Wie in "Fort-Data" und "Festung"), wenn der Speicherplatz knapp wird. Es gibt jedoch Sparmöglichkeiten!

Sie werden vermutlich herausfinden, daß die meisten Zeichen in einem Gemälde 'vollständige Quadrate', d.h. CHR$ 128 oder CHR$ 143, sind. Nicht nur sind diese Zeichen genau gleich, (mit Hinter- und Vordergrundfarben vertauscht), sie entsprechen auch genau CHR$ 32, das der Computer in jedes Element von a$ hineinstellt, wenn es mit DIM aufgebaut wird. Die einzigen Zeichen, die in die Feldgruppe eingefügt werden müssen, sind die 'unvollständigen Quadrate'.

Als ich dieses Programm schrieb, schrieb ich zuerst die Zahlen 129 bis 135 untereinander und zählte dann die Quadrate auf dem karierten Zeichenpapier von 1 bis 704. Jedes Mal, wenn ich ein Zeichen entdeckte, das kein ganzes Quadrat war, schrieb ich die Nummer dieses Quadrates neben den entsprechenden Zeichen-Code.

Diese Liste wurde zu den DATA-Zeilen 129 bis 135 des Programms, und die Zeilen 10 bis 80 wurden hinzugefügt, um die Daten in a$ zu lesen und sie am Schirm auszugeben.

DIE ATTRIBUTE

Die nächste Stufe bestand darin, das karierte Papier mit der

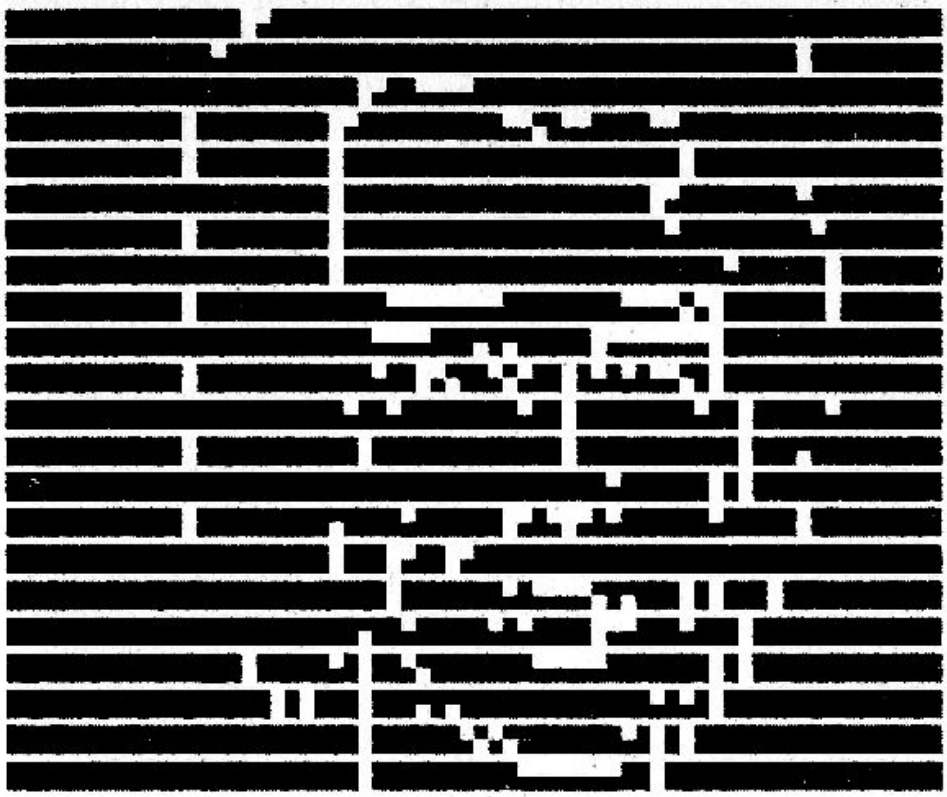
die Farben in jedem Quadrat ankam. Man könnte wiederum 704 Befehle schreiben, um jede Farbkombination in die Attribut-Datei hineinzuPOKEn, aber es gibt auch hier wieder Einsparmöglichkeiten.

Sie werden merken, daß eine Farbe in einem größeren Block gleich bleibt, es ist also einfacher, die Attribute für ein Quadrat aufzuschreiben, gefolgt von der Anzahl der Quadrate, für die die gleichen Attribute gelten.

Die DATA-Zeilen 5000 bis 5021 stellen die Attribute des gesamten Schirms dar, nach der oben erwähnten Methode aufgezeichnet. Beachten Sie, daß Variablennamen anstelle der tatsächlichen Attribute verwendet werden. Die Zeilen 500 bis 570, die die Werte dieser Variablen festlegen, wurden später hinzugefügt, nachdem ich das Programm ATTR TAFEL zu Rate gezogen hatte. Zum Schluß kamen noch die Zeilen 1000 bis 1020 dazu, um die Attributdaten in die Attributdatei einzulesen, und das Bildschirmportrait war fertig.

Genauso wie alle anderen Programme in Kunst um der Kunst willen, ist auch "MARYLIN" als Demonstration gedacht, aber wenn Sie erst einmal die Technik beherrschen, versuchen Sie einmal, Ihre eigene Spectrum Kunstgalerie aufzubauen.

Zeile 1040 gestattet Ihnen, das Bild **OHNE** das Programm, mit dessen Hilfe es entstand, abzuspeichern. (Mit LOAD "MARYLIN"SCREEN$ können Sie das Bild jederzeit reproduzieren). Sie können es aber auch mit dem Maschinenprogramm "BLÄTTERN" abspeichern und wieder aufrufen.



```
  1 BORDER 1: CLS
  5 LET x=9999
 10 DIM a$(704)
 20 FOR j=129 TO 135
 30 RESTORE j
 40 READ a: IF a=x THEN GO TO 7
0
 50 LET a$(a)=CHR$ j
 60 GO TO 40
 70 NEXT j
 80 PRINT a$
129 DATA 57,165,306,331,332,356
,371,456,527,553,555,588,597,617
,618,651,x
130 DATA 197,202,232,308,336,34
0,361,367,373,428,460,462,467,55
9,560,563,595,657,667,x
131 DATA 81,82,106,109,111,266,
267,272,273,274,275,297,298,299,
307,330,525,526,589,590,664,665,
666,x
132 DATA 110,303,304,329,337,38
9,469,523,524,564,594,625,626,65
5,x
133 DATA 37,122,137,149,154,181
,213,218,228,245,260,264,282,296
,328,333,346,359,365,391,397,404
,410,423,424,453,474,501,518,520
,521,531,583,551,584,596,600,616
,628,630,631,649,650,660,682,692
,x
134 DATA 265,335,656,x
135 DATA 24,84,117,170,300,338,
461,463,497,499,556,x
```

```
ET w=56
  510 LET cy=46: LET yc=53
  520 LET my=30: LET ym=51
  530 LET mw=31: LET wm=59
  540 LET yw=55: LET wy=62
  550 LET mr=26: LET rm=19
  560 LET wr=58: LET rw=23
  570 LET gy=38: LET gm=35: LET w
g=60: LET cw=47: LET kw=7: LET k
g=4: LET km=3
1000 RESTORE 5000: FOR j=22528 T
O 23231: READ a,b
1010 FOR k=1 TO b: POKE j,a: LET
 j=j+1: NEXT k: LET j=j-1
1020 NEXT j
1030 IF INKEY$<>"s" THEN GO TO 1
030
1040 SAVE "MARYLIN"SCREEN$
1050 STOP
5000 DATA c,5,y,18,yc,1,c,12
5001 DATA cy,1,y,19,yc,1,c,11
5002 DATA y,12,my,2,m,1,my,1,y,5
,c,11
5003 DATA y,5,my,1,m,2,my,1,ym,1
,my,1,m,5,my,1,y,4,yc,1,c,10
5004 DATA y,4,ym,1,m,11,mw,1,y,4
,yc,1,c,10
5005 DATA cy,1,y,4,ym,1,m,10,mw,
1,y,4,c,11
5006 DATA yc,1,y,4,my,1,m,10,mw,
1,y,4,yc,1,c,9
5007 DATA cy,1,y,3,wy,1,m,12,mw,
1,y,5,c,9
5008 DATA cy,1,y,3,wm,1,km,1,m,1
,km,1,m,4,km,1,m,2,km,1,m,1,w,1,
y,4,yc,1,c,9
5009 DATA y,4,w,1,km,1,gm,2,km,1
,m,2,mw,1,wg,1,g,1,gm,2,w,2,y,4,
c,10
5010 DATA y,4,w,1,kw,1,u,1,kg,1,
g,1,wm,1,m,1,wm,1,kg,1,kw,1,u,1,
k,1,w,2,y,4,yc,1,c,9
5011 DATA cy,1,y,2,yw,1,m,5,wm,1
,m,1,mw,1,m,5,wm,1,y,5,c,10
5012 DATA cy,1,y,1,yw,1,m,5,wm,1
,m,6,mw,1,w,1,y,4,yc,1,c,10
5013 DATA y,2,yw,1,wm,1,m,3,wm,1
,m,7,w,2,y,5,c,10
5014 DATA cy,1,y,2,wm,1,m,3,kw,4
,m,3,wm,1,w,1,wy,1,y,4,yc,1,c,11
5015 DATA y,2,w,1,m,8,km,1,m,1,m
w,1,w,1,wy,1,y,4,c,12
5016 DATA cy,1,y,1,yw,1,wm,1,m,1
,mr,1,rw,1,rm,1,wr,1,rm,1,m,3,mw
,1,w,1,y,5,yc,1,c,12
5017 DATA cy,1,y,1,wm,1,m,1,mr,1
,rw,1,w,2,rw,1,mr,1,m,2,wm,1,wg,
```

```
mw,2,wg,1,gy,1,y,2,yc,1,c,15
5019 DATA cy,1,yw,1,wm,1,m,6,mw,
1,wm,1,m,1,wg,1,g,1,gy,1,yc,1,c,
17
5020 DATA cy,1,yw,1,wm,1,m,3,mw,
1,wm,1,mw,1,m,2,wg,1,g,2,c,10
5021 DATA cw,1,m,1,mw,4,m,4,wg,1
,g,2,c,10
9000 SAVE "MARYLIN" LINE 0
```

MALEN NACH NUMMERN

Dies ist ein viel einfacheres Programm als Marylin – das Bild wird ohne Verwendung der Ausgabedatei aufgebaut. Es ist ausschließlich aus Leerzeichen (Blancs) zusammengesetzt, aber da die Blancs eine unterschiedliche Hintergrundfarbe haben, wird die Illusion eines Bildes erzeugt! (Wenn Sie es nicht glauben wollen, daß das Bild eine Illusion ist, dann versuchen Sie es doch einmal mit COPY über den ZX-Printer auszudrucken).

Alles was Sie tun müssen, ist die Taste zu drücken, deren Nummer mit der Farbe übereinstimmt, die Sie malen wollen.

Das BEEP-Signal sagt Ihnen, wenn der Computer für den nächsten Tastendruck bereit ist.

```
55555775555555555655555555655555556
55557777555656555555577777556566565
22222777775655556557777556556666655
66666277755577755555775566666666666
44444625556777755555555555566666555
11111462577777555555555555656565565
33331462557754455555775556555555565
55565314625556424477777655555544 5
55424531462555544244777765554424 4
54444553146254454445427765554442
44245555314624240504244555545244
24444553146254450544455424440 50
44454455314625555055555544242405
06044245314625555065555544544420 4
50544444314625556055555444050405
50552555314625555055655552550550 5
40444444444444444404444444440440 4
40444444444444444404444444440440 4
40444441111114444444444444440440 4
40444111111111114444444444440444 4
```

```
    1 CLS : PRINT "555557755555555
555555555565555566"
    2 PRINT "555577775556555555557
77755656656565"
    3 PRINT "222227777755555555577
7755556666665"
    4 PRINT "666662777557777555557
755666666666"
    5 PRINT "444446255557777555555
555555666655"
    6 PRINT "111114625777777555555
555556566565"
    7 PRINT "333331462557754455555
775565555556"
    8 PRINT "555553146255554244777
77655555445"
    9 PRINT "554245314625555442447
77775554424 4"
   10 PRINT "544445531462544544454
427755554442"
   11 PRINT "442455553146242405044
24456554524 4"
   12 PRINT "244445553146254450544
44554244405 0"
   13 PRINT "444544553146255550555
55555442440 5"
   14 PRINT "050442453146255550555
55544544420 4"
   15 PRINT "505444443146255550555
55544405040 5"
   16 PRINT "505525553146255550555
55552550550 5"
   17 PRINT "404444444444444440444
44444444044 04"
   18 PRINT "404444444444444440444
44444444044 04"
   19 PRINT "404444411111144444444
44444444044 04"
   20 PRINT "404441111111111144444
44444444044 44"
   21 PRINT "404411111111111111144
44444444044 44"
   22 PRINT "401111111111111111111
11444444044 44"
   50 LET a$="01234567"
  100 BEEP .1,20
  105 LET i$=INKEY$: IF i$="" TH
EN GO TO 105
  110 FOR p=1 TO 8: IF i$=a$(p) T
HEN GO TO 120
  115 NEXT p: GO TO 105
  120 BEEP 1,50: FOR j=0 TO 21: F
OR k=0 TO 31
  125 IF SCREEN$ (j,k)=i$ THEN PR
INT AT j,k; PAPER VAL i$;CHR$ 32
  130 NEXT k: NEXT j
  135 LET a$(p)="*": IF a$="*****
***" THEN STOP
```

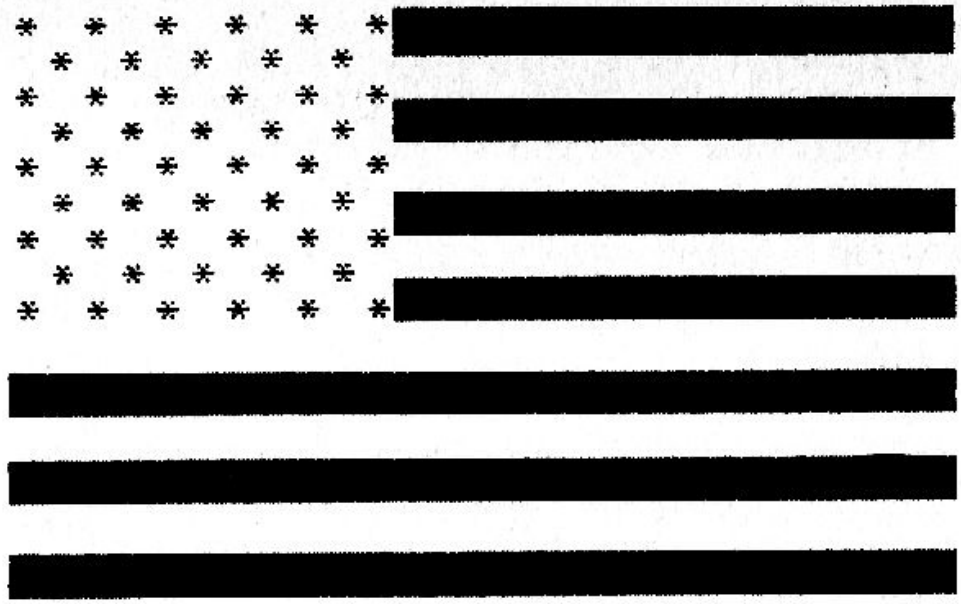
STERNEN-BANNER

Wenn Sie versuchen, die amerikanische Flagge auf dem Bildschirm des Spectrum darzustellen, werden Sie unmittelbar mit einem mathematischen Problem konfrontiert. Der Teil der Fahne mit den Sternen ist neun Zeichen hoch (ich gehe davon aus, daß Sie das Sternchen-Zeichen als Stern verwenden.) Aber angrenzend an diese neun Reihen von Zeichen, gibt es nur sieben Reihen von 'Streifen', und sieben in neun, das geht einfach nicht.

Sie müssen daher an 'Bildpunkte' (Pixels) statt an Zeichen denken. Die Sterne haben eine Tiefe von 9 × 8 = 72 Bildpunkten, nun, sieben geht auch nicht in 72, aber es fällt viel weniger auf, wenn man einen Streifen um 2 Bildpunkte breiter macht, als wenn plötzlich zwei extra Streifen in der Flagge auftauchen.

Deshalb ist der erste Streifen 12 Bildpunkte tief, und alle anderen 10. Man könnte PLOT- und DRAW-Befehle verwenden, um sie zu zeichnen, aber ich habe mich dazu entschieden, die normale Spectrum Graphik zu verwenden plus zwei selbst definierte Graphikzeichen (CHR$ 144 = 6 Reihen von Hintergrund Bildpunkten und 2 Reihen Vordergrund; CHR$ 145 = 2 Reihen Hintergrund (PAPER) und 6 Reihen Vordergrund (INK)).

Die DATA für die selbstdefinierten Graphikzeichen finden sich in Zeile 500. Die DATA-Zeile 1000 enthält die CODEs der Zeichen, die zum Erzeugen der Streifen benutzt werden.



```
    5 BORDER 0: PAPER 7: CLS
   10 RESTORE : FOR j=USR CHR$ 14
4 TO USR CHR$ 144+15: READ a: PO
KE j,a: NEXT j
   15 LET s$="* * * * * *": REM
Zwischenraeume zwischen den Ster
nen
   20 LET t$="  "+s$( TO 10)
   25 PRINT PAPER 1; INK 7;s$'t$'
s$'t$'s$'t$'s$'t$'s$
   30 DIM b$(5)
   35 FOR j=0 TO 16: READ a
   40 FOR k=11*(j<=9) TO 26: PRIN
T PAPER 7*(j<16); INK 2;AT j,k;C
HR$ a;: NEXT k: PRINT PAPER 0;b$
   45 NEXT j: DIM b$(32)
   50 FOR j=0 TO 4: PRINT PAPER 0
;b$: NEXT j
   60 GO TO 60
  500 DATA 0,0,0,0,0,0,255,255,0,
0,255,255,255,255,255,255
1000 DATA 143,131,144,143,128,14
5,131,144,143,128,145,131,144,14
3,128,145,131
9000 SAVE "FLAGGE" LINE 0
```

MELODIEN

Es gibt eine lustige Aufforderung im Spectrum-Handbuch, die gesamte erste Symphonie von Mahler als Programm zu schreiben.

Das ist eine fürchterliche Aufgabe, besonders wenn Sie jede einzelne Note als eigenen BEEP-Befehl eingeben müssen, was die Methode ist, die das Handbuch verwendet.

Es wäre um einiges leichter, wenn Sie DATA-Anweisungen benutzen, um Tonhöhe und -länge von jedem BEEP festzulegen, wie in MELODIE 1.

MELODIE 2 verwendet **getrennte** DATA-Anweisungen für Dauer (Zeilen 100 –) und Höhe (Zeilen 200 –), und die Daten werden in eine Feldgruppe eingelesen, bevor das Lied gespielt wird. Diese Methode erscheint vielleicht plump, aber sie erleichtert das 'Komponieren' ungemein. Sie können sich zunächst für die Tonhöhe entscheiden und fügen den Wert für die Tonlänge erst später ein.
Zufällig ist keine der beiden Melodien von Mahler, aber mit einem bißchen Glück könnten Sie sie erkennen.

```
    5 RESTORE
   10 READ p,d: BEEP d/5,p: GO TO
  10
  100 DATA 12,2,12,2,9,3,9,1,9,2,
5,2,0,2,2,2,5,2,5,2,9,9
  110 DATA 9,1,9,1,9,1,9,2,5,2,5,
22,2,0,2,0,1,0,1,2,2,5,3,7,1,5,1
0
  120 DATA 12,2,12,2,9,2,10,2,9,2
,5,2,0,1,0,1,2,1,5,3,5,2,7,10
  130 DATA 5,2,5,4,2,1,2,1,2,2,0,
2,5,2,5,3,2,1,2,2,0,4,2,2,5,2,7,
2,5,18
9000 SAVE "MELODIE 1"
```

```
   5 DIM p(65): DIM d(65)
  10 RESTORE 100: FOR j=1 TO 65:
READ p(j): NEXT j
  15 RESTORE 200: FOR j=1 TO 65:
READ d(j): NEXT j
  20 FOR k=1 TO 5: FOR j=1 TO 65
  25 BEEP d(j)/5,p(j)
  30 NEXT j
  35 PAUSE 20: NEXT k: STOP
 100 DATA 7,7,7,7,7,7,7,9,7,9,11
 110 DATA 14,14,14,14,14,14,12,1
1,9,9,9,9
 120 DATA 7,7,7,7,7,7,7,7,9,7,9,
11
 130 DATA 14,14,14,14,14,12,11,9
,9
 140 DATA 14,14,14,14,14,14,14,1
4,14,14,14,14,14
 150 DATA 11,12,14,14,12,11,9,7
 200 DATA 2,1,3,2,1,3,2,1,3,2,10
 210 DATA 1,1,2,2,2,1,3,2,2,2,2,
8
 220 DATA 2,2,2,2,2,3,1,2,4,1,1,
10
 230 DATA 1,1,4,2,2,2,2,2,8
 240 DATA 1,1,1,3,2,1,1,2,2,2,2,
2,4
 250 DATA 1,1,6,2,2,2,12,8
9000 SAVE "MELODIE 2"
```

OLDIES

OCHSEN UND KÜHE
HANGDATA/HANGMAN
SIMON SAGT
SKIZZENBLOCK

OCHSEN UND KÜHE

Die Programme in diesem Kapitel sind fast so alt wie die Computer, und sie alle basieren auf Ideen, die so alt sind wie die Menschheit selbst.

Aber die Programme selbst sind neu geschrieben, und zwar speziell für den Spectrum. Außerdem sind sie alle Beispiele dafür, wie ich diese Ideen in Programme umgesetzt habe; verschiedene Programme können zu identischen Lösungen führen, obwohl völlig verschiedene Lösungswege beschritten wurden.

"Ochsen und Kühe" ist das alte 'Codebrecher'-Spiel, das unter unzähligen anderen Namen bekannt ist.

Der Spectrum stellt einen Code von vier beliebigen Farben aus sechs (ausschließlich Schwarz und Weiß) zusammen, wobei alle Farben auch mehrfach auftreten können.
Sie haben zehn Versuche, den Code zu erraten. Verwenden Sie die Tasten 1 bis 6, entsprechend den sechs Farben, um Ihren Versuch einzugeben.

Der Spectrum bewertet jeden Versuch, indem er Ihnen ein schwarzes Quadrat für jede richtige Farbe in der richtigen Position, und ein weißes Quadrat für jede richtige Farbe in der falschen Position zuerkennt.

Nebenbei, dies ist nur ein Geschicklichkeitsspiel. Alle Ihre Versuche und die Bewertung bleiben auf dem Schirm sichtbar, und sie sollten daher immer in der Lage sein, den Code in weniger als 10 Versuchen zu entschlüsseln. In der Tat wurde das Spiel ausgiebig untersucht, und es wurde bewiesen, daß es möglich ist, jeden beliebigen Code in weniger als 5 Versuchen zu knacken!

```
    1 REM Ochsen & Kuehe
    5 RESTORE : FOR j=0 TO 7: REA
D p: POKE USR CHR$ 144+j,p: NEXT
 j
   10 BORDER 7: PAPER 7: INK 9: C
LS
   20 DIM a(4): DIM b(4): DIM c(4
)
   30 LET s=1: RANDOMIZE
   40 FOR j=1 TO 4: LET a(j)=INT
(RND*6)+1: NEXT j
   50 PRINT s;TAB 3;
   60 FOR j=1 TO 4
   70 LET i$=INKEY$
   80 IF i$<"1" OR i$>"6" THEN GO
 TO 70
   90 PRINT INK VAL i$;CHR$ 143;C
HR$ 32;CHR$ 32;
  100 IF i$=INKEY$ THEN GO TO 100
  110 LET b(j)=VAL i$: LET c(j)=a
(j)
  120 NEXT j: PRINT CHR$ 32;CHR$
32
  130 LET p=0
  140 FOR j=1 TO 4
  150 IF b(j)=c(j) THEN GO SUB 50
0
  160 NEXT j
  170 IF p=4 THEN GO TO 700
  180 FOR j=1 TO 4: FOR h=1 TO 4
  190 IF b(j)>0 AND b(j)=c(h) THE
N GO SUB 600
  200 NEXT h: NEXT j
  210 PRINT : PRINT
  220 IF NOT p THEN BEEP .5,2: GO
 TO 240
  230 FOR j=p TO 3: BEEP .1,5: NE
XT j
  240 LET s=s+1: IF s<=10 THEN GO
 TO 50
  300 PRINT "Du hast den Code nic
ht geknackt"
  310 PRINT TAB 3;: FOR j=1 TO 4:
 PRINT INK a(j);CHR$ 143;CHR$ 32
;CHR$ 32;: NEXT j
  320 INPUT "P um weiterzuspielen
: X um auf- zuhoeren ";a$
  330 IF a$="p" THEN RUN
  340 IF a$="x" THEN STOP
  350 GO TO 320
  500 PRINT CHR$ 143;CHR$ 32;: BE
EP .01,50: LET b(j)=0: LET c(j)=
0: LET p=p+1: RETURN
  600 PRINT CHR$ 144;CHR$ 32;: BE
EP .01,30: LET b(j)=0: LET c(h)=
0: LET p=p+1: RETURN
  700 PRINT AT s*2,9; INK 2; FLAS
H 1;"RICHTIG BEI  ";s
  710 GO TO 320
```

HANG-DATA/-HANGMAN

Versuchen Sie mir nicht einzureden, Sie hätten noch kein Hangman-Programm. Es gibt eines im Spectrum-Handbuch!

Der Hauptunterschied besteht darin, daß diese Version eine 'Datei' mit Wörtern besitzt, so daß Sie das Spiel für einige Zeit spielen können, ohne daß Sie eine zweite Person benötigen, die Ihnen neue Wörter zum Raten eingibt.

"Hangdata" ist das Programm zum Erstellen der Datei. Sie können so viele Wörter eingeben wie Sie wollen, die Grenzen liegen bei der Speichergröße des Spectrum und bei Ihrer Geduld. Ich argwöhne, daß die letztere das Limit setzen wird.

Die Wörter sollten nicht länger als 32 Zeichen sein, aber sie können Leerzeichen, Bindestriche, oder, wenn Sie richtig gemein sein wollen, auch Graphikzeichen enthalten.

Um Probleme mit großen und kleinen Buchstaben zu vermeiden, enthält "Hangdata" jedoch eine Routine, um kleine Buchstaben in große umzuwandeln (Zeile 70). "Hangman" enthält ähnliche Routinen (Zeilen 110 und 180).

Wenn Sie Ihre Wörter eingegeben haben, werden sie auf Band gespeichert, um automatisch durch das Programm "Hangman" geladen zu werden. Es wäre daher ideal, wenn auf Ihrem HANGMAN-Band unmittelbar nach dem Programm "Hangman" eine durch "Hangdata" erzeugte Datei folgen würde. Es gibt jedoch keine Begrenzung für die

eine gute Idee, eine Bibliothek von Dateien zu erstellen, mit verschiedenen Schwierigkeitsgraden für verschiedene Altersgruppen.

```
   10 PRINT "Wieviele Woerter wil
lst Du ein- geben?"
   20 INPUT n: CLS
   30 DIM w$(n+1,33): LET w$(1)=S
TR$ n
   40 FOR j=2 TO n+1
   50 INPUT ("Eingabe Wort #";j-1
;":");t$
   60 LET l=LEN t$: IF NOT l OR l
>32 THEN GO TO 50
   70 FOR k=1 TO l: IF t$(k)>="a"
 AND t$(k)<="z" THEN LET t$(K)=C
HR$ (CODE t$(k)-32)
   80 IF t$(k)<CHR$ 32 THEN GO TO
 50
   90 NEXT k
  100 LET w$(j)=CHR$ l+t$
  110 PRINT t$: NEXT j
  120 SAVE "data" DATA w$()
  130 STOP
9000 CLEAR : SAVE "HANGDATA" LIN
E 0
```

HANGMAN

Diese Version ähnelt so sehr der alten 'Papier-und-Bleistift'-Version, daß nur wenige Worte zur Erklärung nötig sind.

Der 'Mann' und der 'Galgen' werden Strich für Strich gezeichnet, und Sie sind 'tot', wenn die Zeichnung beendet ist. (11 falsche Versuche). Es gibt keine ausgearbeitete 'Todesroutine' wie im Spectrum-Handbuch!

Die beiden DATA-Zeilen (1000 und 1010) enthalten die PLOT- und DRAW-Koordinaten für die Zeichnung. Für jede Linie werden je zwei PLOT- und DRAW-Koordinaten benötigt, außer für den Kopf, der – welche Überraschung – durch den CIRCLE-Befehl in Zeile 2000 gezeichnet wird.

Wenn es Ihnen mißlingt, das Wort zu erraten, dann blinken die fehlenden Buchstaben auf dem Bildschirm (Zeilen 510–530).

Wenn Sie alle eingegebenen Begriffe verwendet haben, dann können Sie entweder:

– RUN eingeben und eine neue Begriffsdatei laden

oder

– GO TO 10 eingeben und die gleichen Begriffe – in geänderter Reihenfolge – erneut benutzen.

```
    5 LOAD "data" DATA w$()
   10 BORDER 4: PAPER 4: INK 9
   15 LET n=VAL w$(1)
   20 DIM w(n): LET u=1
   25 CLS : RESTORE
   30 IF INKEY$="p" OR INKEY$="P"
 THEN GO TO 30
   35 LET w=INT (RND*n)+2
   40 FOR j=1 TO u: IF w(j)=w THE
N GO TO 35
```

```
+1
   50 LET t$=w$(w,2 TO CODE w$(w,
1)+1): LET l=LEN t$
   55 FOR j=1 TO l: PRINT AT 0,j;
CHR$ 45: NEXT j: LET j=0
   60 LET r=0
  100 LET i$=INKEY$: IF i$<CHR$ 3
2 THEN GO TO 100
  110 IF i$>="a" AND i$<="z" THEN
  LET i$=CHR$ (CODE i$-32)
  120 LET x=1: FOR k=1 TO l
  130 IF t$(k)=i$ THEN PRINT AT 0
,k;i$: LET x=0: LET t$(k)=CHR$ 0
: LET r=r+1
  140 NEXT k
  150 IF x THEN LET j=j+1: GO SUB
  2000
  160 IF j=11 THEN GO TO 500
  170 IF r=l THEN GO TO 600
  180 IF INKEY$=i$ OR INKEY$=CHR$
  (CODE i$+32) THEN GO TO 180
  190 GO TO 100
  500 PRINT AT 15,15; FLASH 1; IN
K 2; PAPER 7;"FALSCH"
  510 FOR k=1 TO l
  520 IF t$(k)<>CHR$ 0 THEN PRINT
  AT 0,k; FLASH 1; INK 2; PAPER 7
;t$(k)
  530 NEXT k
  540 IF u=n+1 THEN PRINT AT 21,2
;"ALLE EINGEGEBENEN WORTE VER-
   WENDET": GO TO 9999
  550 PRINT AT 21,0;"DRUECKE P UM
  EIN NEUES SPIEL     ZU SPIELEN :
  ODER X UM AUFZU-    HOEREN"
  560 IF INKEY$="p" OR INKEY$="P"
  THEN GO TO 25
  570 IF INKEY$="x" OR INKEY$="X"
  THEN GO TO 9999
  580 GO TO 560
  600 PRINT AT 15,15; FLASH 1; IN
K 2; PAPER 7;"GESCHAFFT"
  610 GO TO 540
1000 DATA 16,40,48,0,40,40,0,72,
40,112,24,0,64,112,0,-10,40,100,
12,12
1010 DATA 64,96,6,64,90,0,-24,64
,66,-8,-12,64,66,8,-12,64,82,-12
,2,64,82,12,2
2000 READ a,b: IF j=6 THEN READ
c: CIRCLE a,b,c: GO TO 2020
2010 PLOT a,b: READ c,d: DRAW c,
d
2020 RETURN
9000 CLEAR : SAVE "HANGMANN" LIN
E 0
9999 STOP
```

SIMON SAGT

In der althergebrachten Version dieses Spiels mußte man alles wiederholen, was 'Simon' tat, wodurch oft ziemlich groteske Situationen entstehen konnten.

Die Computerversion ist sehr viel 'würdevoller'. 'Simon' (Spectrum) präsentiert Ihnen eine immer länger werdende Folge von Farben und Sie müssen diese Folge wiederholen, unter Verwendung der Farbtasten Ø bis 7.

Zu jeder Farbe gehört ein spezieller Ton, und sie wird an der Stelle des Schirms angezeigt, die der entsprechenden Tastenposition entspricht. Wenn Sie das Spiel wirklich schwierig machen wollen, dann versuchen Sie, es auf einem Schwarz-Weiß-Fernseher zu spielen. (Mein bestes Ergebnis war dabei 5).

Bei einem Farbfernseher ist es natürlich möglich, erheblich höhere Ergebnisse zu erzielen. Allerdings ist das Spiel mehr als ein einfacher Gedächtnistest, da die Spielgeschwindigkeit, mit wachsender Länge der Folge, erheblich zunimmt.

Unterprogramm 1000 erzeugt die Farbanzeige und die Töne.

Die 'Zeitvariable' t ist umgekehrt proportional zur Länge der Farb- und Tonfolge.

Wenn (falls!) Sie einen Fehler machen, dann wiederholt Zeile 510 die Folge, nachdem sie t auf einen konstanten Wert von 0,8 Sekunden gesetzt hat, durch Aufrufen des Unterprogramms ab Zeile 1005 anstelle von Zeile 1000.

```
    1 REM SIMON SAGT
    5 BORDER 7: PAPER 7: INK 9: B
RIGHT 0: CLS : DIM b$(32)
   10 RANDOMIZE
   15 LET r$=""
   20 LET x=INT (RND*8)
   25 LET r$=r$+STR$ x
   30 LET n=LEN r$
   35 FOR j=1 TO n
   40 LET x=VAL r$(j)
   45 GO SUB 1000
   50 NEXT j
  100 PRINT AT 20,0;"Jetzt wieder
holen Sie die Folge"
  105 FOR j=1 TO n
  110 LET i$=INKEY$
  115 IF i$<"0" OR i$>"7" THEN GO
 TO 110
  120 LET x=VAL i$
  125 GO SUB 1000
  130 IF INKEY$=i$ THEN GO TO 130
  135 IF i$<>r$(j) THEN GO TO 500
  140 NEXT j
  145 PRINT AT 20,0;b$;AT 20,0;"
bisher geschafft : ";n
  150 FOR j=1 TO 100: NEXT j
  155 CLS : GO TO 20
  500 PRINT AT 20,0; INK 9; FLASH
  1;"IRRTUM!!!!! Die Folge war
    "
  505 FOR j=1 TO 250: NEXT j
  510 LET t=.8: FOR j=1 TO n: LET
  x=VAL r$(j): GO SUB 1005: NEXT
j
  515 PRINT AT 20,0;b$;AT 20,0;"S
ie erreichten ";n-1'"Druecken Si
e""R"" zum weitermachen"
  520 IF INKEY$<>"r" THEN GO TO 5
20
  530 RUN
1000 LET t=1/n
1005 PRINT AT 6,x*3+30*(x=0); IN
K x; BRIGHT 1;CHR$ 143;: BEEP t,
x*7+56*(x=0): PRINT CHR$ 8;CHR$
32
1010 RETURN
9000 SAVE "SIMON SAGT" LINE 0
```

SKIZZEN-BLOCK

Warum den Computer nicht als Skizzenblock verwenden? Diese Programme machen Spaß, und Sie können einige echt gute Bilder mit der hochauflösenden Graphik des Spectrum entwerfen.

In diesem Programm zeigt ein 'Punktcursor' die gegenwärtige Zeichen-(DRAW)-Position an. Die 'Pfeiltasten' 5 bis 8 veranlassen, daß eine Linie in der angezeigten Richtung gezeichnet wird. Sie können den Cursor bewegen ohne zu zeichnen, indem Sie auf Taste "1" drücken, wodurch der Cursor blinkt, und dann die Tasten 5-8 wie vorher verwenden. In diesem Modus durchquert der Cursor auch früher gezeichnete Linien ohne sie zu löschen. Um weiterzeichnen zu können, drücken Sie auf "0", und Sie erhalten wieder den normalen Cursor.

Zum Löschen einer Zeile drücken Sie "0" (Sie werden jetzt annehmen, daß der Spectrum knapp an Tasten ist, weil ich zwei völlig verschiedene Funktionen auf die '0-Taste' gelegt habe), und die Richtungstasten werden jetzt löschen. Zur Rückkehr in den normalen Modus wieder "0" drücken.

Ein interessanter Punkt zum Merken - das hat nichts mit "Skizzenblock" zu tun, aber das ist das einzige Programm, bei dem ich diese Technik angewendet habe:
Wenn Sie die Bedienungsanweisungen für ein Programm in eine REM-Zeile am Ende dieses Programms schreiben, dann können Sie diese Zeile während des Programmlaufs LISTen und dann im Programm weitermachen (siehe Zeile 1). Ich weiß nicht, ob das irgendwelche Vorteile gegenüber einem PRINT-Befehl hat, aber es ist etwas, was die meisten anderen Beispiele nicht tun, und deshalb sollte ich es

```
    1 REM Skizzenblock
    2 BORDER 7: INK 0: PAPER 7: C
LS : LIST 9999
    5 IF INKEY$<>"r" THEN GO TO 5
   10 LET beweg=0: LET halt=0
   15 PAPER 0: BORDER 0: INK 7: C
LS
   20 LET x1=0: LET y1=0
   25 LET x=0: LET y=0
   30 LET x1=x+x1: LET y1=y+y1
   35 IF INKEY$="0" THEN LET bewe
g=NOT beweg: LET halt=0
   40 IF INKEY$="1" THEN LET bewe
g=1: LET halt=1
   45 IF INKEY$="0" THEN GO TO 45
   50 LET x=((INKEY$="8")*(x1<255
))-((INKEY$="5")*(x1>0))
   55 LET y=((INKEY$="7")*(y1<175
))-((INKEY$="6")*(y1>0))
   60 IF halt=1 AND POINT (x1,y1)
=1 THEN GO TO 30
   65 PLOT OVER 0;x1,y1
   70 IF beweg=1 THEN PLOT OVER 1
;x1,y1
   75 GO TO 30
9000 SAVE "SKIZZEN" LINE 0
9999 REM Benutze die Pfeiltasten
 zum Malen
           Benutze Taste 1 um den
Cursor zurueckzustellen
           Benutze Taste 0 zum
Loeschen Benutze Taste 0 zum
Abschalten der Loesch/Rueckstell
funktion
        Druecke R zum Starten
```

Lösung zu der Aufgabe auf Seite 104.

PROGRAMM LÖSUNG

```
    5 CLS
   10 FOR x=70 TO 160 STEP 30
   15 FOR y=50 TO 140 STEP 30
   20 PLOT x,y
   25 NEXT y: NEXT x
   30 PRINT AT 20,3;"Druecke irge
ndeine Taste fuer die Loesung":
PAUSE 0
   40 PLOT 160,140
   50 RESTORE : FOR j=1 TO 6: REA
D a,b: DRAW a,b: PAUSE 20: NEXT
j: STOP
  100 DATA -60,0,0,-90,90,0,-120,
120,0,-150,90,90
9000 SAVE "LOESUNG" LINE 0
```

SAM, SPIEL ES IMMER WIEDER . . .

SKYLINE
SPIESSRUTENLAUF
UFOS
FEUERKRAFT

SKYLINE

Sie sind der Kommandant eines außerirdischen Raumschiffs, das auf der Erde landen möchte. Nachdem Sie eine der großen, modernen Städte wie London oder New York ausgesucht haben, in der überaus optimistischen Hoffnung, Sie können dort intelligentes Leben vorfinden, machen Sie Ihren Anflug.

Sie übersehen vollständig, daß es in der Nähe große, gut ausgebaute Flughäfen gibt, und fliegen direkt ins Zentrum der Stadt.

Die Wolkenkratzer könnten ein Problem für Sie bedeuten, aber Sie nehmen an, daß es sich dabei nur um große Schutthaufen handelt (ein ganz natürlicher Fehler) und beschließen, den Erdlingen einen Gefallen zu tun, indem Sie sie einebnen und eine Landebahn bauen.

Es gibt drei Möglichkeiten, wie Sie einen Wolkenkratzer zerstören können (immer nur ein Stockwerk auf einmal):

1) mit Bomben (Taste "Ø")
2) mit Raketen (Taste "9")
3) indem Sie ihn mit Ihrem Schiff rammen.

Diese letzte Methode ist nicht empfehlenswert, denn Sie haben nur vier Schiffe, mit denen Sie eine erfolgreiche Landung durchführen sollen.

Das Hauptziel des Spieles ist es, alle Gebäude einzuebnen, um dadurch einem Ihrer Schiffe die Landung zu ermöglichen. Punkte sind nur von untergeordneter Bedeutung. Was nützt Ihnen ein hohes Punktekonto, wenn alle Ihre Schiffe zerstört sind? Andererseits können Sie nur dann eine Rakete abfeuern, wenn Sie einen positiven Punktestand haben.

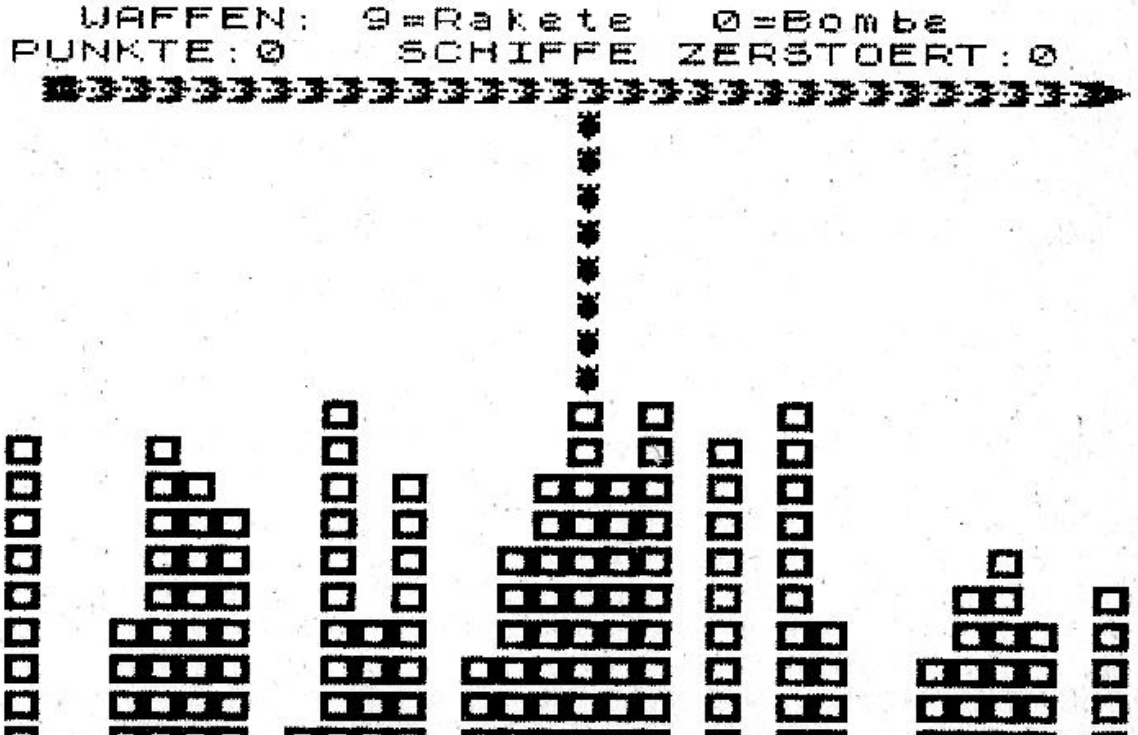
1 Punkt für jedes Stockwerk eines Wolkenkratzers, das durch eine Bombe zerstört wurde.
-20 Punkte, wenn Sie eine Rakete verwenden
-20 Punkte, wenn Sie ein Schiff verlieren

Bonuspunkte für eine erfolgreiche Landung:

400 für eine Landung mit dem ersten Schiff
300 für eine Landung mit dem zweiten Schiff, usw.

Ihre wichtigste Waffe sind daher Bomben. Sie fallen immer senkrecht, aber sie können natürlich danebenfallen, was Zeit kostet.

Raketen sind teure Waffen und sollten nur in Notfällen benutzt werden (und auch nur dann, wenn Sie einen positiven Punktestand haben, damit Sie sie überhaupt anwenden können). Raketen werden waagrecht abgeschossen und treffen IMMER. Wenn kein Wolkenkratzer direkt vor Ihnen ist, dann fliegen Sie 'hinten' um den Bildschirm, bis Sie vielleicht einen treffen. Der Abstieg Ihres Schiffes ist 'eingefroren', während eine Rakete in der Luft ist, Sie könnten sie also auch einmal verwenden, um eine Atempause zu erhalten und sich Ihre weitere Vorgehensweise zu überlegen.



```
    1 DATA 0,255,127,255,106,255,
127,255
    2 DATA 0,192,240,252,191,252,
240,192
    3 DATA 42,62,28,62,62,62,28,8
    4 DATA 255,255,195,195,195,19
5,255,255
    5 DATA 0,0,132,198,255,198,13
2,0
   10 FOR g=1 TO 5: RESTORE g
   15 FOR j=0 TO 7
   20 READ a: POKE USR CHR$ (g+14
3)+j,a
   25 NEXT j
   30 NEXT g
   40 LET s$=CHR$ 32+CHR$ 144+CHR
$ 145: LET b$=CHR$ 146
   50 RANDOMIZE
  100 BORDER 5: PAPER 0: INK 6: C
LS
  105 DIM t$(11,32)
  110 FOR j=11 TO 21
  115 IF j>11 THEN LET t$(j-10)=t
$(j-11)
  120 FOR k=1 TO 4: LET t$(j-10,(
INT (RND*32)+1))=CHR$ 147: NEXT
k
  125 PRINT INK 7;AT j,0;t$(j-10)
  130 NEXT j
  135 PRINT AT 0,2; INK 7;"WAFFEN
: 9=Rakete  0=Bombe"
  200 LET sc=0: LET sl=0
  205 LET x=sl*2+2: LET y=0: LET
bf=0
  210 PRINT AT 1,0; INK 4;"PUNKTE
:";sc;TAB 11;"SCHIFFE ZERSTOERT:
";sl
  215 PRINT AT x,y;$$
  220 IF INKEY$="0" AND bf=0 THEN
 LET bf=1: LET bx=x: LET by=y: B
EEP .01,25
  225 IF bf THEN IF x=20 THEN LET
 bf=INT (RND*2)
  230 IF INKEY$="9" AND sc>0 THEN
 GO TO 2000
  235 IF bf THEN GO SUB 1000
  300 LET y=y+1
  305 IF y=32 THEN LET y=0: LET x
=x+1
  310 IF x=21 AND y=30 THEN GO TO
 5000
  350 IF x<11 THEN GO TO 210
  355 LET a=x-10: LET b=y+3: IF b
>32 THEN LET b=b-32: LET a=a+1
  360 IF t$(a,b)<>CHR$ 147 THEN G
O TO 210
  365 FOR k=7 TO 0 STEP -1: PRINT
```

```
 370 LET t$(a,b)=CHR$ 32
 375 LET sl=sl+1: LET sc=sc-20
 380 IF bf=1 THEN PRINT AT bx,by
;CHR$ 32
 385 IF sl=4 THEN GO TO 4000
 390 GO TO 205
1000 PRINT AT bx,by;CHR$ 32
1005 IF bx=21 THEN GO TO 1030
1010 LET bx=bx+1
1015 IF bx<11 THEN GO TO 1025
1020 IF t$(bx-10,by+1)=CHR$ 147
THEN GO TO 1100
1025 PRINT INK 2;AT bx,by;b$
1030 IF bx=21 THEN BEEP .05,5: P
RINT AT bx,by;CHR$ 32: LET bf=0
1040 RETURN
1100 LET sc=sc+1
1105 LET bf=0: LET t$(bx-10,by+1
)=CHR$ 32: PRINT AT bx,by;CHR$ 3
2
1110 BEEP .01,35
1115 RETURN
2000 LET a=x: LET b=y+3: IF b>31
 THEN LET b=b-32: LET a=a+1
2005 LET sc=sc-20
2010 IF a>=11 THEN GO TO 2040
2015 PRINT AT x,y; INK 1+(b/2=IN
T (b/2));s$
2020 PRINT AT a,b; INK 3;CHR$ 14
8
2025 BEEP .01,40
2030 PRINT AT a,b;CHR$ 32
2035 LET b=b+1: IF b=32 THEN LET
 b=0: LET a=a+1
2040 IF a<11 THEN GO TO 2015
2045 IF t$(a-10,b+1)=CHR$ 147 TH
EN LET t$(a-10,b+1)=CHR$ 32: PRI
NT AT a,b;CHR$ 32: BEEP 1,35: GO
 TO 210
2050 IF a=21 AND b=31 THEN GO TO
 210
2055 GO TO 2015
4000 PRINT AT 10,5; INK 2; PAPER
 7; FLASH 1;"ALLE SCHIFFE VERLOR
EN"
4005 GO TO 5010
5000 PRINT AT 10,15; INK 2; PAPE
R 7; FLASH 1;"SIEG": LET sc=sc+1
00*(4-sl)
5005 PRINT AT 1,0; INK 8; PAPER
8; FLASH 1;"PUNKTE ";sc
5010 PRINT AT 15,5; PAPER 7; INK
 2;"Druecke 'P' zum Weiterspiele
n";AT 17,10;"oder 'X' zum AUfhoe
ren"
5015 IF INKEY$="x" THEN GO TO 50
30
```

```
5030 PAPER 5: INK 0: CLS : PRINT
 AT 10,15;"BYE": STOP
9000 SAVE "SKYLINE" LINE 0
```

ANMERKUNGEN ZUM PROGRAMM

Die vier selbstdefinierten Graphikzeichen sind:

CHR$ 144 und CHRS$ 145:	Ihr Schiff
CHR$ 146 :	Bombe
CHR$ 147 :	Ein Wolkenkratzer-'Stock-werk'

Die 'Silhouette' wird durch die Zeilen 105 bis 130 erstellt. Die Feldgruppe t$( ) enthält die Position der einzelnen Hochhäuser, und wenn Stockwerke zerstört werden, dann werden die Werte in der Feldgruppe entsprechend verändert. Das Programm liest nicht den Bildschirm-speicher, was beim Spectrum grundsätzlich nicht möglich ist, es benutzt auch nicht ATTR oder SCREEN$, um die Anwesenheit eines Häuserblocks festzustellen, sondern es vergleicht lediglich die Position von Schiff/Bombe/Rakete mit dem entsprechenden Element von t$.

Die Hauptschleife des Programms, nämlich das Herunter-gehen des Schiffes, liegt zwischen den Zeilen 200 und 360 (oder es geht bis 250, wenn das Schiff in der oberen Hälfte des Schirms, oberhalb der Wolkenkratzer fliegt). Die Zeilen 365 bis 390 behandeln den Absturz eines Schiffes.

Unterprogramm 1000 zeichnet die Bombe in der jeweili-gen Position, wobei die Zeilen 1100–1115 den Treffer an-zeigen.

Unterprogramm 2000 zeigt die Rakete an, genau auf die gleiche Weise wie die Schiffs-Landeschleife, und es kann erst nach einem Treffer verlassen werden (Zeile 2050). (Beachten Sie: Zeile 2045 ist eine Art 'Idiotensicherung'. Es gibt absolut keinen Grund, Punkte zu verschenken, indem man eine Rakete abfeuert, wenn bereits alle Hochhäuser zerstört sind, aber falls irgend jemand das tut, dann wird die Routine durch diese Zeile beendet, sobald die Rakete

Wie kann man das Spiel einfacher/schwieriger machen?

Die Anzahl der Hochhäuser wird durch das Limit von k in Zeile 120 kontrolliert. k=1 TO 3 macht die Abstände größer, k=1 TO 5, macht sie kleiner. Zeile 225 hindert den 'Bombenknopf' daran, daß er jedesmal funktioniert, wenn Sie gerade zur Landung ansetzen. Dadurch kommt ein wenig 'Torschlußpanik' ins Spiel, da Sie gewzungen sein könnten, Raketen einzusetzen, um ein paar Erdgeschosse zu zerstören. Wenn Sie das zu frustrierend finden, können Sie diese Zeile komplett löschen.

Um die Anzahl der Schiffe zu verändern, müssen Sie Zeile 385 ändern. Um die Punktevergabe zu verändern, schauen Sie in den Zeilen 375 (Absturz), 1100 (Bomben), 2005 (Raketen) und 5000 (Bonus) nach.

SPIESS-RUTENLAUF

"Spießrutenlauf" ist ein Spiel, das ich ein "Kriegsspiel für Pazifisten" nennen würde. Sie werden ununterbrochen angegriffen, aber Sie selbst sind nicht in der Lage, den Gegner in irgend einer Weise anzugreifen. Sie müssen versuchen, die Niederlage durch die Anwendung Ihrer verschiedenen Verteidigungsmechanismen abzuwenden.

Ihr Raumschiff startet oben links am Schirm und bewegt sich im Zick-zack nach unten, bis es schließlich unten links am Bildschirm landet. (Beachten Sie die Richtungsumkehr im Gegensatz zum 'hinten herum fliegen' bei "Skyline").

Der Gegner bedient die Laserkanonen am Fuß des Schirms. Ein Treffer zerstört nicht direkt Ihr Schiff, aber er vermindert Ihre Energiereserven, die der Schlüsselwert in diesem Spiel sind. Sie beginnen mit 1000 Energieeinheiten, und diese werden eine nach der anderen verbraucht, während Sie den Bildschirm überqueren. Sie können zusätzliche Einheiten bei den 'Stationen mit dem grünen Kreuz' an verschiedenen Stellen während Ihres Abstiegs nachladen – wenn Sie also den Laserstrahlen ausweichen können, dann sollten Sie in der Lage sein, sicher zu landen.

VERTEIDIGUNGSMECHANISMEN

Die folgenden Einrichtungen verleihen Ihnen alle einen gewissen Schutz gegen die Laser, aber sie alle verbrauchen Energie!

Taste	Energieverbrauch	Wirkung
Ø	5 Einheiten	Vermindert die wirksame Reichweite der Laser
9	10 Einheiten	Vergrößert die zeitlichen

8 20 Einheiten

Laserschüssen
Erhöht die Geschwindigkeit
Ihres Schiffes

```
    1 RESTORE : FOR j=0 TO 23: RE
AD a: POKE USR (CHR$ 144)+j,a: N
EXT j
    5 BORDER 0: PAPER 0: INK 7: C
LS
   10 RANDOMIZE
   15 LET e=1000: PRINT "RESTENER
GIE ";e
   20 LET a=2: LET b=0
   25 FOR j=3 TO 18 STEP 3: PRINT
 AT j,0; INK 4;CHR$ 146;AT j,31;
 INK 4;CHR$ 146;AT 21,4*j/3; INK
 3;"↑": NEXT j
   35 LET s$=CHR$ (144+(a/2<>INT
(a/2)))
   40 IF e<0 THEN LET e=0
   45 PRINT AT 0,12;e;CHR$ 32;CHR
$ 32: IF e=0 THEN GO TO 500
   50 PRINT AT a,b; INK 6;s$
   55 LET t=7: IF INKEY$="9" THEN
 LET e=e-10: LET t=4*t+RND*10
   60 LET l=144: IF INKEY$="0" TH
EN LET e=e-5: LET l=INT (RND*144
)+1
   65 GO SUB 1000
   70 LET an=a: LET bn=b
   75 IF INKEY$="8" THEN LET e=e-
20: LET bn=bn-2+4*(s$=CHR$ 144)
   80 LET bn=bn-1+2*(s$=CHR$ 144)
: IF bn>31 OR bn<0 THEN LET an=a
n+1: LET bn=31*(s$=CHR$ 144): IF
 an=20 THEN GO TO 550
   85 PRINT AT a,b;CHR$ 32
   90 LET e=e-1: IF ATTR (an,bn)=
4 THEN LET e=e+INT (RND*100)+75:
 BEEP .1,55
 100 LET a=an: LET b=bn: GO TO 3
5
 500 PRINT AT 3,9; INK 2; FLASH
1;"SCHIFF ZERSTOERT"
 510 FOR j=7 TO 0 STEP -1: PRINT
 AT a,b; INK j; BRIGHT 1;"*": BE
EP .5,j*8: NEXT j
 520 GO TO 600
 550 PRINT AT 3,1; INK 4; FLASH
1;"SICHER GELANDET.GUT GEMACHT"
 600 INPUT "Druecke P zum weiter
spielen,      X zum Aufhoeren ";s$
 610 IF s$="p" THEN RUN 15
 620 IF s$="x" THEN GO TO 9999
 630 GO TO 600
1000 LET x=INT (RND*t)+1: IF x>6
```

```
1005 LET x=32*x+4
1010 PLOT INK 2;x,9
1015 LET c=0: DRAW OVER 1; INK 2
;0,l: IF ATTR (a,b)<>6 THEN LET
c=1
1020 BEEP .01,l/3
1025 IF c THEN PRINT FLASH 1; PA
PER 2;AT a,b;"*": BEEP 1,20: LET
 e=e-INT (RND*50)-100
1030 PLOT OVER 1; INK 2;x,9: DRA
W OVER 1; INK 2;0,l
1035 RETURN
5000 DATA 0,224,112,255,112,224,
0,0,0,7,14,255,14,7,0,0,60,60,25
5,255,255,255,60,60
9000 SAVE "GAUNTLET" LINE 0
9999 STOP
```

ANMERKUNGEN ZUM PROGRAMM

Das Programm verwendet zwei selbstdefinierte Graphikzeichen, eines für ein Schiff, das nach rechts fliegt, und das andere für ein Schiff, das nach links fliegt.

Die schnellen Laserschüsse werden durch die DRAW-Befehle im Unterprogramm 1000 erzeugt.

Ein Treffer wird gelöscht durch die ATTR-Funktion in Zeile 1015. Das Schiff wird in Farbe 6, der Laserstrahl in Farbe 2 angezeigt; deshalb ändert der Laserstrahl die Attribute der Schiffsposition von 6 auf 2, wenn ein Treffer erzielt wird.

Die Länge eines Laserstrahles wird durch die Variable 1 bestimmt (normalerweise 144 Bildpunkte, aber reduziert, wenn INKEY$ = "0", Zeile 60).

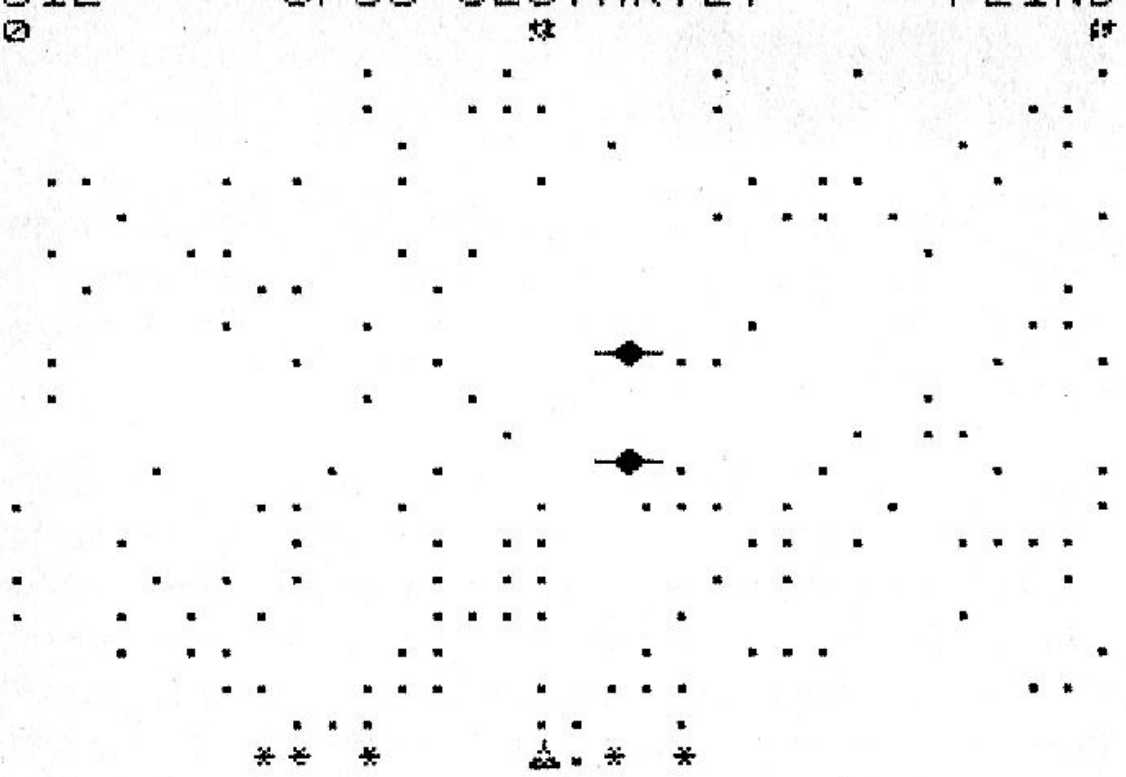
Welcher Laser gerade feuert, wird durch die Zufallszahl x (Zeile 1000) bestimmt, und wenn x größer ist als 6, dann fällt kein Schuß. Die Zufallszahl liegt innerhalb der Grenzen 1 bis t, d.h., je größer der Wert von t wird, um so geringer ist die Wahrscheinlichkeit, daß ein Schuß abgefeuert wird. t ist ursprünglich auf 7 (für fast ununterbrochene Schußfolge) gesetzt und wird vergrößert, wenn INKEY$ = "9" ist (Zeile 55).

UFOS

In "Skyline" hatten Sie alle Waffen, und der Verteidiger hatte keine. In "Spießrutenlauf" hatte der Verteidiger alle und Sie nichts. Ist es nicht allmählich an der Zeit, für einen ausgeglicheneren Wettstreit?

In diesem Spiel bedienen Sie die Raketenstationen auf der Erde, aber anstatt passiv darüber herumzufliegen, werfen die gegnerischen Untertassen pausenlos Bomben auf Sie. Was noch schlimmer ist: Während Ihre Schüsse senkrecht nach oben geschossen werden müssen, haben die Untertassen die unangenehme Fähigkeit, diagonal auf Sie feuern zu können. Sie setzen auch von Zeit zu Zeit 'unsichtbare' UFOS ein, die nicht leicht zu treffen sind, die Ihnen aber Bonuspunkte einbringen. Und um es noch schwieriger zu machen, sind die 'Ringe' der Untertassen 'ätherisch', Sie müssen also einen direkten Treffer auf dem Mittelteil des Flugkörpers landen, um Punkte zu erhalten. Andererseits ist Ihre Basis sehr verwundbar, und Sie können leicht seitwärts in Bomben hineinlaufen, die Sie ansonsten verfehlt hätten.

Ihre Steuertasten sind "8" und "9", um nach links und rechts zu steuern, und "0" zum Feuern. Das Spiel ist beendet, wenn 50 Untertassen erfolgreich über Sie hinweggeflogen sind, und Sie gewinnen, wenn Sie bis dahin die Mehrzahl der Treffer erzielt haben. Sie brauchen nicht zu verzweifeln, wenn Sie meilenweit zurückliegen, und nur noch wenige Ufos kommen. Jedes Mal, wenn Sie einen Treffer landen, wird der Ufozähler nicht erhöht, und eine neue Untertasse wird auf dem gleichen Kurs gestartet, wie die, die Sie gerade getroffen haben. Mit etwas Geschick ist es so möglich, in einen 'Rhythmus' zu kommen und eine große Zahl von Abschüssen zu erzielen, bevor einer Untertasse ein erfolgreicher Vorbeiflug gelingt.




```
    1 REM Fliegende Untertassen
    2 RESTORE : FOR j=USR CHR$ 14
4 TO USR CHR$ 144+23: READ a: PO
KE j,a: NEXT j
    5 BORDER 1: PAPER 1: INK 0: C
LS
   10 FOR k=1 TO 10: GO SUB 8500:
 NEXT k
   15 LET s$=CHR$ 32+CHR$ 145+CHR
$ 146
   20 LET bf=0: LET mf=0: LET h=0
: LET x=15: LET s=0: LET t=0: LE
T ns=0
   25 PRINT AT 0,0; PAPER 5;"SIE"
;TAB 8;"UFOS GESTARTET";TAB 27;"
FEIND"'s;TAB 15;ns;TAB 31;t
   30 GO SUB 8600: GO TO 1000
  100 LET x1=x+(INKEY$="9" AND x<
31)-(INKEY$="8" AND x>0)
  110 IF x<>x1 THEN PRINT AT 21,x
;CHR$ 32;AT 21,x1; INK 6;CHR$ 14
4: LET x=x1
  120 IF INKEY$="0" AND mf=0 THEN
 LET mf=1
  130 RETURN
1000 GO SUB 8500
1005 LET a=INT (RND*10): LET b=I
NT (RND*10): LET a=17-a: LET i=I
NT (RND*7)+1
1010 LET ns=ns+1: PRINT AT 1,15;
 PAPER 6;ns: IF ns>50 THEN GO TO
 7000
1015 IF bf=0 THEN LET bf=1: LET
bx=a+1: LET y=SGN (x-b): LET by=
```

```
1025 PRINT AT a,b; INK i,s$
1030 IF bf THEN GO SUB 5000
1035 IF mf THEN GO SUB 6000
1040 GO SUB 100
1045 LET b=b+1: IF b=30 THEN PRI
NT AT a,30;CHR$ 32;CHR$ 32: GO T
O 1000
1050 GO SUB 100
1055 GO TO 1015
5000 PRINT AT bx,by;CHR$ 32: LET
 bx=bx+1: LET by=by+y
5005 IF by<0 OR by>31 THEN LET b
y=by-y
5010 IF bx=22 THEN GO TO 5050
5015 PRINT AT bx,by;".": RETURN
5050 LET h=0: IF ABS (by-x)<=1 T
HEN LET by=x: LET h=1: LET t=t+1
: PRINT AT 1,32-LEN STR$ t; PAPE
R 8;t
5055 LET bx=bx-1: FOR j=7 TO 1 S
TEP -1: PRINT AT bx,by; INK j;"*
": NEXT j
5060 BEEP h+.01,50
5065 LET bf=0: RETURN
6000 IF mf=1 THEN LET mx=20: LET
 my=x
6005 PRINT AT mx,my;CHR$ 32: LET
 mx=mx-1: IF SCREEN$ (mx,my)=CHR
$ 46 OR mx=2 THEN GO TO 6050
6010 PRINT AT mx,my;"†"
6015 IF mx<>a THEN GO TO 6040
6020 IF my<>b+2 THEN GO TO 6040
6025 LET s=s+1: FOR j=6 TO 1 STE
P -1: PRINT AT a,b; INK j;"****"
: NEXT j: BEEP .5,30
6030 IF i=1 THEN LET s=s+4
6035 PRINT AT 1,0; PAPER 8;s: LE
T mf=-1: LET b=0
6040 LET mf=mf+1: RETURN
6050 LET mf=0: IF mx<>2 AND ATTR
 (mx,my)=8 THEN LET bf=0
6055 PRINT AT mx,my;CHR$ 32
6060 RETURN
7000 PRINT AT 1,12; FLASH 1; INK
 2; PAPER 7;"SPIELENDE": LET w=s
7005 IF t>s THEN LET w=t
7010 PRINT AT 1,(32-LEN STR$ t)*
(t=w); FLASH 1; PAPER 7; INK 0;w
7015 IF t=s THEN PRINT AT 1,0; F
LASH 1; PAPER 7; INK 0;s
7020 INPUT "Druecken Sie ""ENTER
""um neu zu beginnen";s$
7025 RUN 10
8000 DATA 8,8,28,28,62,62,127,12
7,1,3,7,255,7,3,1,0,128,192,224,
255,224,192,128,0
8500 FOR j=1 TO 10: PRINT AT INT
```

```
8600 PRINT AT 21,x; INK 6;CHR$ 1
44: RETURN
9000 SAVE "UFOS" LINE 0
```

ANMERKUNGEN ZUM PROGRAMM

Die selbstdefinierten Graphikzeichen sind:

CHR$ 144 : Raketenbasis
CHR$ 145 und CHR$ 146 : Fliegende Untertasse.

Die Hauptschleife des Programms liegt zwischen Zeile 1015 und Zeile 1055. Untertassen werden an den Koordinaten a,b, die beide als Zufallszahlen in Zeile 1005 bestimmt werden, gestartet.

Die Farbe (INK) der Untertassen ist ebenfalls vom Zufall bestimmt und eigentlich unwichtig, außer daß beispielsweise eine blaue Zeichnung auf blauem Hintergrund die 'unsichtbaren' Untertassen entstehen läßt.

Unterprogramm 100, das zweimal in der Hauptschleife aufgerufen wird (wie in "Übungsspiel") bewegt die Raketenbasis.

Unterprogramm 100 erzeugt neue 'Sterne' (siehe unten)
Unterprogramm 5000 zeichnet Ihre Raketen.
Unterprogramm 6000 zeichnet die Bomben.

VORGEHENSWEISE

Die Sterne in diesem Spiel sind sehr wichtig und mehr als nur ein hübscher Hintergrund für die Handlung.

Sie können immer nur eine Rakete auf einmal in der Luft haben, deshalb geht Zeit verloren, während Sie darauf warten, daß die Rakete trifft oder den oberen Bildrand erreicht. Sterne dienen als nützliche 'Ziele', da Sie Ihre vorbeigegangenen Raketen zerstören und Ihnen so zu einem neuen Schuß verhelfen. (Wenn Sie einen Stern treffen, erhalten Sie natürlich keine Punkte, und der Stern wird

zwischen Ihnen und den Untertassen Sie beim Treffen Ihres eigentlichen Ziels. Wenn Sie strategisch richtig spielen, dann sollten Sie versuchen, im oberen Teil des Himmels eine 'Decke' aus Sternen zu erhalten und jene in der Nähe des Bodens zu zerstören. Es lohnt sich oft in der Anfangsphase eines Spiels, die Untertassen zu ignorieren (aber auf die Bomben aufzupassen) und einfach zu versuchen, ein günstiges 'Sternenmuster' zu erstellen.

Auf den Gegner allerdings haben die Sterne keinen Einfluß, und sowohl Ufos als auch Bomben durchfliegen sie ungestraft. Die Sterne werden zerstört, wenn sie durchquert werden, aber das könnte sich zu Ihren Gunsten auswirken. Denken Sie außerdem daran, daß jedes Mal, wenn eine neue Untertasse gestartet wird, gleichzeitig bis zu zehn neue Sterne entstehen.

FEUERKRAFT

Zum Schluß eine völlig andere Art von Weltraumspiel.

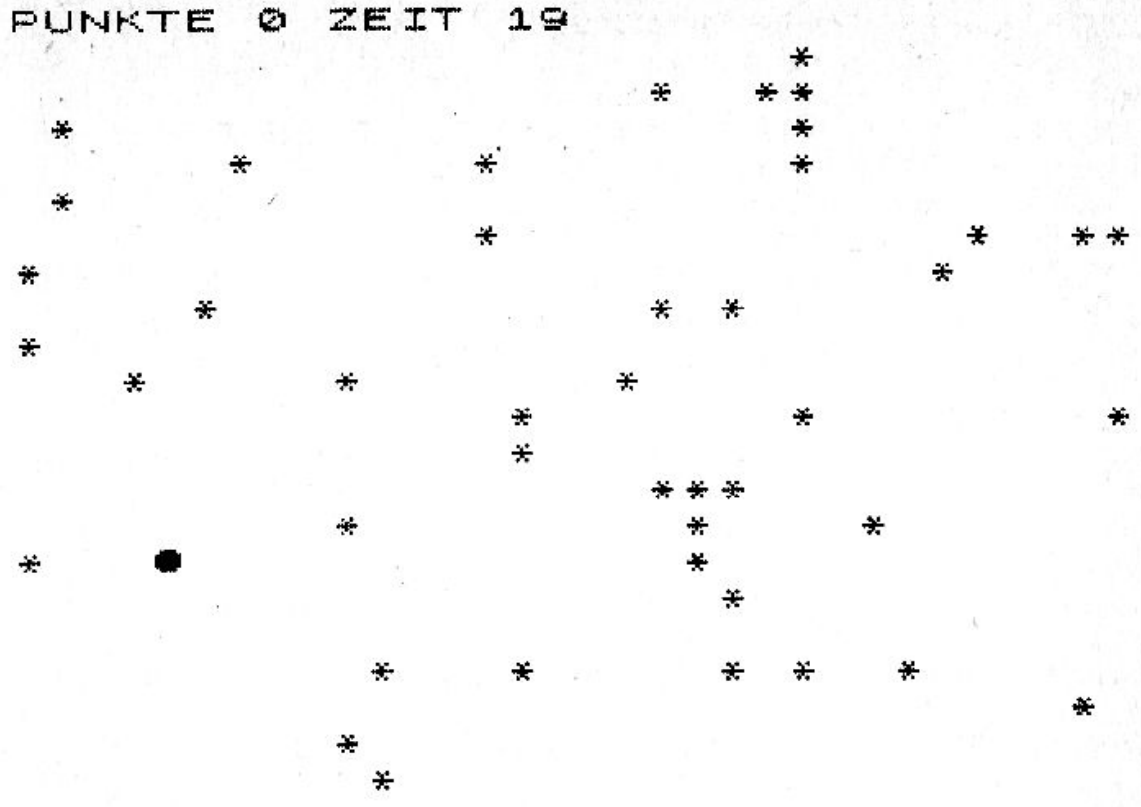
In den vorangegangenen drei Spielen konnten Sie auf dem Bildschirm wie im Lehnstuhl die Handlung beobachten. In "Feuerkraft" sitzen Sie selbst an den Kontrollhebeln der Laserkanone Ihres Raumschiffs, und der Bildschirm zeigt Ihnen den Blick durch das Visier.

Sie haben 60 Sekunden, um so viele Feindschiffe wie möglich abzuschießen, indem Sie sie genau ins Zentrum des Visierkreuzes nehmen (2 Punkte), oder indem Sie einen der 'Fäden' mit dem Raumschiff zur Deckung bringen (1 Punkt).

Ein 'Fadentreffer' zerstört den Gegner **nicht**, und es sollte deshalb möglich sein, bei jedem senkrecht anfliegenden Schiff drei Punkte zu erzielen. (Zuerst den oberen 'Faden' zur Deckung bringen und dann, unmittelbar danach, den Gegner ins Zentrum des Fadenkreuzes hineinfliegen lassen).

Bei diagonal anfliegenden Schiffen können zwei Punkte erzielt werden, entweder durch einen direkten Treffer oder mit zwei erfolgreichen Streifschüssen. (Die erste Methode ist vorzuziehen, da das Schiff auch nach dem zweiten Treffer noch nicht zerstört ist und Sie darauf warten müssen, bis es den unteren Bildrand erreicht).
In diesem Spiel sind Sie NICHT schneller als der Gegner, also ist das Stellungsspiel von äußerster Wichtigkeit. Jagen Sie nicht aussichtslos hinter weit entfernten Schiffen her, sondern konzentrieren Sie sich darauf, eine günstige Position zum Abfangen des nächsten zu erreichen.

Benutzen Sie die Tasten "5", "6", "7" und "8", um zu zielen. Drücken Sie auf die "0"-Taste, wenn Sie einen 'Streifschuß' erzielen wollen. Wenn Sie den Gegner im Zentrum des Visierkreuzes haben, feuert Ihr Laser automatisch.




```
    9 FOR j=USR CHR$ 144 TO USR C
HR$ 144+7: READ a: POKE j,a: NEX
T j
   10 DIM a$(65): LET a$(1)=CHR$
124: LET a$(32)=CHR$ 45: LET a$(
34)=CHR$ 45: LET a$(65)=CHR$ 124
   15 BORDER 5: PAPER 5: INK 0: O
VER 1: CLS
   20 FOR j=1 TO 50: PRINT AT INT
 (RND*22),INT (RND*32);"*": NEXT
 j
   25 LET s=0
   30 LET x=11: LET y=15
   35 POKE 23673,0: POKE 23672,0
   40 LET t=0
   45 LET b=0
  100 LET t1=INT ((PEEK 23673*256
+PEEK 23672)/50)
  105 IF t<>t1 THEN BEEP 0.1,t: L
ET t=t1
  110 PRINT AT 0,0; OVER 0;"PUNKT
E ";s;" ZEIT ";t
  115 IF t>=60 THEN GO TO 1000
  120 IF NOT b THEN LET b=INT (RN
D*31)+1: LET a=1: LET d=INT (RND
*3)-1: GO TO 130
  125 PRINT AT a,b;CHR$ 144: LET
a=a+1: LET b=b+d
  130 IF a=21 OR b=0 OR b=32 THEN
 LET b=0: GO TO 200
  135 PRINT AT a,b; INK 6;CHR$ 14
4
  200 PRINT AT x,y; INK 6;a$
  205 LET p=(x+1)*22+y: LET p1=a*
22+b: IF p=p1 THEN BEEP 1,50: LE
```

```
LET b=0: GO TO 220
 210 IF INKEY$<>"0" THEN GO TO 2
20
 215 IF ABS (p-p1)=1 OR ABS (p-p
1)=22 THEN BEEP .5,50: LET s=s+1
 220 LET x1=x+(INKEY$="6" AND x<
19)-(INKEY$="7" AND x>0)
 225 LET y1=y+(INKEY$="6" AND y<
30)-(INKEY$="5" AND y>1)
 230 PRINT AT x,y;a$
 235 LET x=x1: LET y=y1
 240 GO TO 100
1000 PRINT OVER 0; FLASH 1; PAPE
R 2; INK 7;AT 10,12;"SPIELENDE";
 FLASH 0;AT 21,2;"DRUECKE ""R""
ZUM WEITERSPIELEN"
1005 IF INKEY$="r" THEN RUN
1010 GO TO 1005
5000 DATA 0,60,126,126,126,126,6
0,0
9000 SAVE "FEUERKRAFT" LINE 0
```

ANMERKUNGEN ZUM PROGRAMM

Das feindliche Schiff ist das einzige selbstdefinierte Graphikzeichen in diesem Programm.
Das 'Visier' wird durch die vier Zeichen in der Feldgruppe a$( ) gebildet. Der Rest der Feldgruppe ist leer, und wenn die gesamte Feldgruppe am Bildschirm ausgegeben wird, entsteht dadurch das 'Fadenkreuz'.

Die Zeiteinteilung wird in den Zeilen 35 (zum Initialisieren), 100 und 105 vorgenommen. Wie in allen Programmen, welche die Systemvariable FRAMES für genaue Zeitmessung verwenden, sollte der Divisionsfaktor in Zeile 100 den gleichen Wert wie die örtliche Stromfrequenz haben. (siehe Anmerkung auf Seite 16).

ZX-SPECTRUM-POWER

Ob Sie nun als Neuling oder schon als versierter Programmierer an den ZX SPECTRUM herangehen, fasziniert von den fantastischen Grafik-, Farb- und Speichermöglichkeiten des SPECTRUMs können Sie nun endlich mit diesem Buch sofort in die Feinheiten einsteigen.

Viele Programme und eine Reihe von Routinen, die Ihnen sehr nützlich sein werden, enthält der vorliegende Band. Hier ein kleiner Ausschnitt aus dem Inhalt: Computerspiele mit beweglicher Grafik, ernsthafte Anwendungen und Geschäftsprogramme, eine Auswahl von Maschinenprogrammen in mnemonischen und Dezimalcode, eine Aufstellung von Unterprogrammen, die Sie eigenen Programmen anfügen können und Weltraumspiele und, und...

Wenn Sie alle Qualitäten Ihres Computers voll ausschöpfen wollen - brauchen Sie dieses Buch.

ZX POWER

ISBN 3-19-008200-6 Hueber Software Cooperation

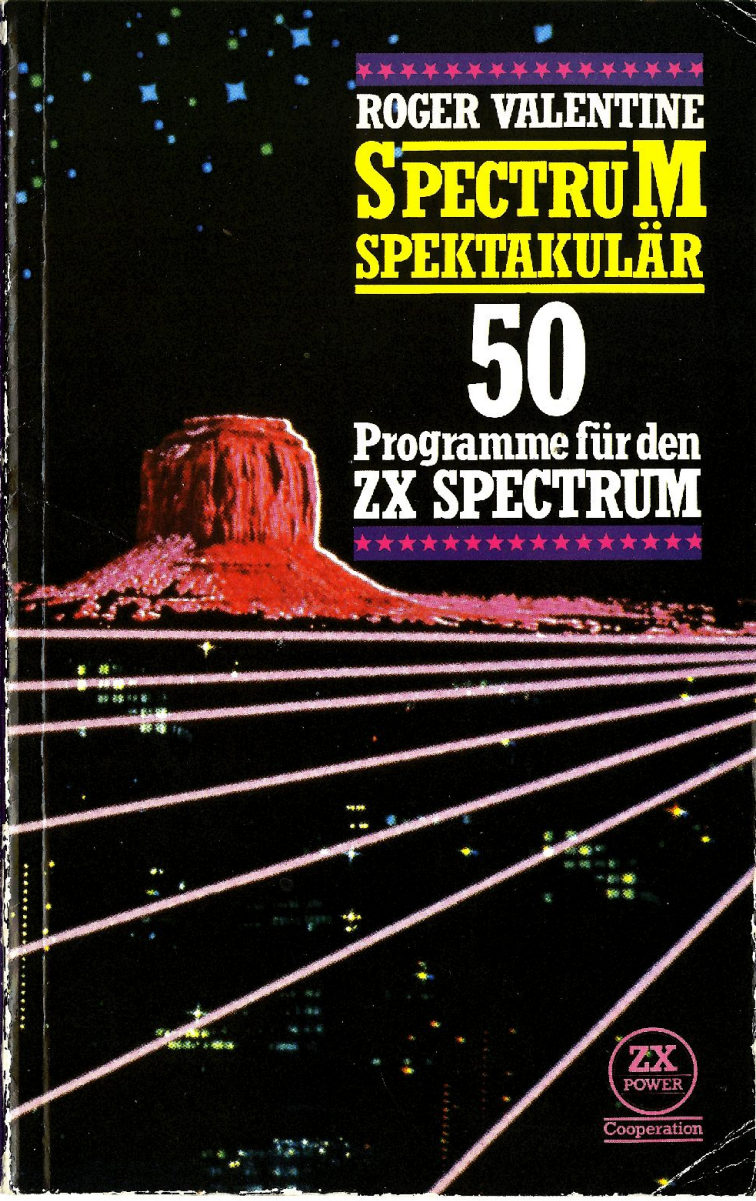
ROGER VALENTINE SPECTRUM SPEKTAKULÄR

ROGER VALENTINE

SPECTRUM SPEKTAKULÄR

50

Programme für den ZX SPECTRUM



ZX POWER Cooperation