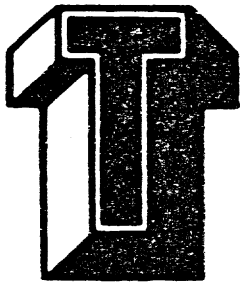
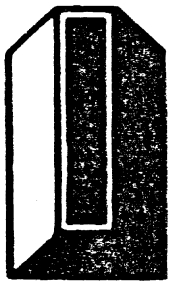
Kleinstrechner



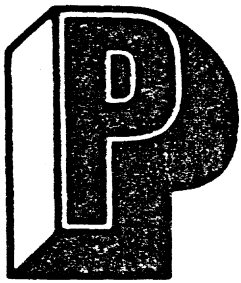
Tendenzen
und Theorien

Grafikbaugruppe
für Z 1013



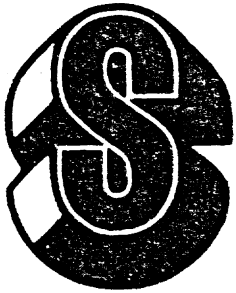
Informationen
und Ideen

Chemisches
Rechnen



Programme
und Projekte

Schiebefax



Spaß
und Spiel

Kleinstrechner-TIPS

Heft 11

Mit 43 Bildern

Herausgegeben von
Prof. Dr.-Ing. Hans Kreul
Prof. Dr. sc. techn. Thomas Horn
Doz. Dr.-Ing. Wilhelm Leupold

VEB Fachbuchverlag Leipzig

Inhalt

ISBN 3-343-00481-2

Zum Inhalt

Das vorliegende Heft 11 der »Kleinstrechner-TIPS« ist entsprechend dem Hauptanliegen dieser Broschürenreihe auf Probleme der Anwendung und Programmierung von Kleincomputern orientiert, wobei es, um den vielfältigen Ansprüchen besser gerecht zu werden, eine relativ breite Auswahl solcher Programmierungs- und Anwendungsprobleme enthält, die in erster Linie für Schulen, Arbeitsgemeinschaften und Computerclubs von Interesse sein dürften. Außerdem wird auf zwei interessante Hardwareerweiterungen für Kleincomputer verwiesen.

LÖHR bringt ein Programm zur Analyse von Schwingungen mit dem Kleincomputer KC 85/2. Das Problem der mathematischen Modellierung von mechanischen und elektrischen Schwingungsvorgängen und ihrer grafischen Darstellung auf dem Kleincomputer wird vor allem für unsere Oberschüler von großem Interesse sein.

LORENZ und SCHULZE schließen mit dem Teil 3 zum Aufsatz »Simulation auf Mikrorechnern — Spiele und Experimente« die in den Heften 6 und 7 begonnene Veröffentlichung ab. Insbesondere geben sie einige Anregungen zur grafischen Darstellung der Simulationsergebnisse und zur Bildanimation, die mit der weiteren Leistungsentwicklung der Mikrorechentechnik zunehmend an Bedeutung gewinnt.

Auch der Beitrag von GIRLICH beschäftigt sich mit der Erzeugung bewegter Bilder auf dem Kleincomputer. Er wendet die Methode des schnellen Umspeicherns des Bildwiederholspeichers an und gibt dafür ein Programm, das als Beispiel die »sich drehende Erde« demonstriert.

KÖHLER beschreibt den schaltungstechnischen Aufbau zum Anschluß einer elektronischen Schreibmaschine S 6005 als Drucker an einen Kleincomputer einschließlich des dazugehörigen Druckertreibers.

RÖBENACK und HOBOHM stellen eine Grafikbaugruppe für den Z 1013 vor, die eine Auflösung von 256 × 256 Bildpunkten ermöglicht.

Auch das Programm von BOCKHACKER zum chemischen Rechnen dürfte für Oberschüler und Lehrer interessant sein, da es unmittelbar Teile des Lehrplans der 7. und 8. Klasse im Fach Chemie auf dem Kleincomputer realisiert.

Im abschließenden Beitrag von GÖRGENS wird der Aufsatz zur Spielprogrammierung, der im Heft 10 begonnen wurde, mit den Spielen »Schiebefax«, »Barrikade« und »Regelmäßige Vielecke« abgeschlossen.

Abschließend verbleibt uns nur, Ihnen viel Spaß bei der Lektüre dieses Heftes zu wünschen.

Thomas Horn

Elementare Behandlung von Schwingungen mit dem Kleincomputer KC 85/2

1. Problemstellung

Zur grafischen Behandlung von Schwingungsvorgängen ist ein BASIC-Programm zu erstellen, welches die Lösung der auftretenden Differentialgleichungen durch elementare Ansätze umgeht. Damit wird das Programm auch für Schüler verständlich, die über keine entsprechenden mathematischen Vorkenntnisse verfügen.

2. Mathematisch-physikalische Grundlagen

2.1. Lösungsverfahren

Als bekannt werden vorausgesetzt die Definitionen von Geschwindigkeit und Beschleunigung in der Form

$$v = \frac{\Delta s}{\Delta t} \tag{1}$$

bzw.

$$a = \frac{\Delta v}{\Delta t}. \tag{2}$$

Ein Massenpunkt besitze zum Zeitpunkt t eine Geschwindigkeit v und den Ort s. Welche Geschwindigkeit und welcher Ort ergeben sich zum Zeitpunkt $t + \Delta t$?
Für kleine Zeitintervalle Δt läßt sich die Bewegung des Massenpunktes in guter Näherung als gleichförmige Bewegung auffassen:

$$t_{neu} = t_{alt} + \Delta t \tag{3}$$

$$\begin{aligned} s_{neu} &= s_{alt} + \Delta s \\ &= s_{alt} + v \cdot \Delta t \text{ wegen (1)} \end{aligned} \tag{4}$$

$$\begin{aligned} v_{neu} &= v_{alt} + \Delta v \\ &= v_{alt} + a \cdot \Delta t \text{ wegen (2).} \end{aligned} \tag{5}$$

Dieser Ansatz, welcher auf den Mathematiker L. Euler (1707 bis 1783) zurückgeht, läßt sich universell anwenden, weil für die Beschleunigung a jede beliebige, physikalisch sinnvolle Funktion eingesetzt werden kann.
Stehen die Beschleunigungswerte für verschiedene Schwingungsvorgänge zur Verfügung, können aus vorstehendem Ansatz heraus durch Iteration (d.h. ständige Wiederholung) die Ort-Zeit-Werte der Schwingung berechnet werden. Die Genauigkeit der Näherung ist von der Wahl des Zeitintervalls Δt abhängig. Zur Durchführung von iterativen Berechnungen ist, bedingt durch die große Anzahl von Einzelrechnungen, der Computereinsatz äußerst effektiv.

2.2. Freie Schwingungen

Wird ein an einer Feder aufgehängtes Massenstück aus seiner Ruhelage ausgelenkt, so führt das System Feder plus Massenstück nach dem Loslassen *freie* Schwingungen aus. Dabei ist die Beschleunigung der Masse, also auch die auf sie wirkende Kraft, die Rückstellkraft, dem Abstand s der Masse von der Ruhelage unter bestimmten Vor-

aussetzungen (HOOKEsches Gesetz) proportional.
Es gilt:

$$F = m \cdot a = -k \cdot s. \quad (6)$$

k Federkonstante in $N \cdot m^{-1}$
m Masse in kg
s Elongation in m

Die Beobachtung zeigt, daß die ausgelösten Schwingungen allmählich unter stetem Kleinerwerden der Amplitude aufhören, so daß letztlich wieder die Ruhelage erreicht wird. Die Schwingungen verlaufen *gedämpft*. Die Abnahme der Amplituden erfolgt durch die Wirkung von Kräften, die der Bewegung des Systems entgegenwirken, z.B. Luftwiderstand und Reibung innerhalb des Federmaterials. Es wird der Fall betrachtet, daß die diese Dämpfung hervorrufende Reibungskraft der Geschwindigkeit der schwingenden Masse proportional ist.

$$F_r = r \cdot v \quad (7)$$

r Reibungskonstante in $kg \cdot s^{-1}$

Da die Reibungskraft hemmend auf die Schwingung einwirkt, gilt für die Kräfte am freien, gedämpften Federschwinger

$$F = m \cdot a = -k \cdot s - r \cdot v. \quad (8)$$

Um die gedämpften Schwingungen systematisieren zu können, wird die bekannte Gleichung für die Schwingungsdauer ungedämpfter Schwingungen herangezogen und unter Berücksichtigung der Reibung entsprechend ergänzt. Aus

$$T = 2\pi \sqrt{\frac{m}{k}}$$

folgt

$$f = \frac{1}{2\pi} \sqrt{\frac{k}{m}}$$

bzw.

$$\omega_0 = \sqrt{\frac{k}{m}} \quad (9)$$

(Kreisfrequenz einer ungedämpften Schwingung)

$$\omega_d = \sqrt{\frac{k}{m} - \frac{r^2}{4m^2}} \quad (10)$$

(Kreisfrequenz einer gedämpften Schwingung).
Der Ausdruck unter der Wurzel, die Diskriminante, ermöglicht eine Fallunterscheidung. Mit

$$D = \frac{k}{m} - \frac{r^2}{4m^2}$$

gilt

$D > 0$ Schwingfall
$D = 0$ aperiodischer Grenzfall
$D < 0$ (aperiodischer) Kriechfall.

2.3. Erzwungene Schwingungen

Bisher wurde der Fall betrachtet, daß die Auslösung der Schwingung durch einen einmaligen Anstoß erfolgt, worauf das schwingungsfähige System *freie* Schwingungen ausführt. Sehr wichtig und interessant ist der Fall, daß die Erregung ebenfalls durch ein schwingendes System erfolgt; es entstehen *erzwungene* Schwingungen. Die äußere Kraft wird dann erfaßt durch die Gleichung

$$F_{err} = F_a \cdot \sin(2\pi f_e \cdot t). \quad (11)$$

Für die Kräfte am Federschwinger, der eine erzwungene gedämpfte Schwingung ausführt, gilt dann

$$F = m \cdot a = F_a \cdot \sin(2\pi f_e \cdot t) - k \cdot s - r \cdot v \quad (12)$$

F_a Amplitude der äußeren Kraft
f_e Frequenz der äußeren Kraft.

Nach Division durch die Masse m steht damit ein Ausdruck für die Beschleunigung a zur Verfügung, mit dem die vorstehend erläuterten Schwingungsvorgänge erfaßt werden:

$$a = a_0 \cdot \sin(2\pi f_e \cdot t) - \frac{k}{m} s - \frac{r}{m} v. \quad (13)$$

2.4. Elektrische Schwingungen

Durch Verdeutlichen der Analogien zwischen den relevanten mechanischen und elektrischen Größen läßt sich ein der Gleichung (13) entsprechender Ansatz für die Darstellung der Vorgänge im L-C-Schwingkreis ableiten.

Mechanische Größen

Elongation $x = x\,(t)$

Geschwindigkeit $v = \frac{\Delta s}{\Delta t}$

Masse m

Kehrwert der Federkonstanten k^{-1}

Rücktreibende Kraft $F = k \cdot x$

Kreisfrequenz des ungedämpften Systems $\omega = \sqrt{\frac{k}{m}}$

Reibungskonstante r

Kreisfrequenz des gedämpften Systems

$$\omega = \sqrt{\frac{k}{m} - \frac{r^2}{4\,m^2}}$$

Äußere Kraft, Erregerschwingung

$F_{err} = F_a \cdot \sin\,(2\,\pi\,f_e \cdot t)$

Elektrische Größen

Ladung $q = q\,(t)$

Stromstärke $i = \frac{\Delta q}{\Delta t}$

Induktivität L

Kapazität C

Spannung am Kondensator $u_C = \frac{1}{C}\,q.$

$$\omega = \sqrt{\frac{1}{L \cdot C}}$$

Ohmscher Widerstand des Schwingkreises R

Erregerspannung

$u_{err} = U_a \cdot \sin\,(2\,\pi\,f_a \cdot t)$

Damit ergeben sich in Analogie zu Gleichung (12) die Beziehungen

$$U = L\frac{\Delta i}{\Delta t} = U_a \cdot \sin\,(2\,\pi\,f_a \cdot t) - \frac{1}{C} \cdot q - i \cdot R \quad \text{bzw.} \qquad (12')$$

$$U = u_{err} - u_C - u_R \qquad (12'')$$

u_C Momentanwert der Spannung am Kondensator

u_R Momentanwert der Teilspannung an R

Daraus folgt nach Division von (12′) durch die Induktivität L der Ausdruck

$$b = \frac{U_a}{L} \sin\,(2\,\pi f_a \cdot t) - \frac{1}{L \cdot C} \cdot q - \frac{R}{L} \cdot i, \qquad (13')$$

aus dem sich die entsprechenden Programmveränderungen ableiten lassen.

Während bei der Betrachtung der mechanischen Schwingungen Elongation und Geschwindigkeit in Abhängigkeit von der Zeit von besonderem Interesse sind, werden bei der Gewinnung von Aussagen über elektrische Schwingungen allgemein die Stromstärke- und die Spannung-Zeit-Werte am Kondensator herangezogen. Das Programm (s. S. 8) ist auf die grafische Darstellung der Elongation in Abhängigkeit von der Zeit für mechanische Schwingungen beschränkt. Es läßt sich jedoch auf die Darstellung der Geschwindigkeit-Zeit-Werte erweitern und für die Untersuchung der elektrischen Schwingungen entsprechend modifizieren. Die dazu erforderlichen Gleichungen wurden bereitgestellt bzw. sind der gegebenen Analogiebetrachtung zu entnehmen.

3. Programmaufbau

Das Programm ermöglicht die grafische Darstellung freier ungedämpfter und gedämpfter sowie erzwungener Schwingungen durch Eingeben folgender Parameter:

Federkonstante k in $\mathrm{N \cdot m^{-1}}$

Masse m in kg

Reibungskonstante r in $\mathrm{kg \cdot s^{-1}}$

Amplitude der äußeren Kraft F_a in N

Frequenz der äußeren Kraft f_e in Hz.

Im Eingabeteil (170···300) werden zusätzlich als Informationen über den je-

weils erfaßten Schwingungsvorgang dessen Eigenfrequenz und der Betrag der Rückstellkraft (230···250) ausgegeben.
Damit sind Bezüge zu den Werten möglich, die als Parameter für die äußere Kraft gewählt werden. In Zeile 240 erfolgt die Definition der Zeitdifferenz für den Fall, daß für die Diskriminante in der Gleichung

$D \leqq 0$

gilt.

In den Zeilen 310···470 wird die Verzweigung zu den unterschiedlichen Schwingungsformen vorgenommen, deren Bezeichnung im Kopffenster ausgegeben wird.
Der entscheidende Programmabschnitt ist der Schleifenteil 520···550, in dem die Momentanwerte für Elongation und Geschwindigkeit berechnet werden. In Zeile 510 erfolgen der Ansatz für die cos-Funktion und die Festlegung der Zeitdifferenz für $D > 0$.

Zur grafischen Darstellung von 2 Perioden der betrachteten Schwingung in einem 2. Fenster dient der Programmteil 560···650; das Koordinatensystem wird durch das Unterprogramm 1000 bereitgestellt. Nach Darstellung von jeweils 2 Perioden wird vom Programmnutzer eine Entscheidung gefordert: entweder Überschreiben des bereits entstandenen Graphen während der nächsten beiden Perioden für Vergleichszwecke oder Darstellung des weiteren Verlaufs der Schwingung in einem neuen Koordinatensystem.
Das Unterprogramm 2000 stellt eine Anregung dar zum Schaffen einer Abfragemöglichkeit zur Dauer des Einschwingvorgangs. Die Einschwingzeit wird bezogen auf die Schwingungsdauer der Erregerschwingung (äußere Kraft) ermittelt und ausgegeben. Wurde der Programmlauf zu diesem Zweck mit ⟨BRK⟩ ⟨GOSUB 2000⟩ unterbrochen, so ist er mit ⟨CONT⟩ fortzusetzen. Dieser Programmabschnitt sollte von Interessenten weiter ausgebaut werden!

4. Liste der verwendeten Variablen

K	Federkonstante	
M	Masse	
C = K/M		
FU	Eigenfrequenz	des ungedämpften Systems
TU	Schwingungsdauer	des ungedämpften Systems
B	Reibungskonstante	
FA	Amplitude	der äußeren Kraft
FE	Frequenz	der äußeren Kraft
R = B/M		
DI = C — R²/4	Diskriminante in OM	
OM	Kreisfrequenz der gedämpften Schwingung	
TD	Schwingungsdauer des gedämpften Systems	
X	Elongation	
D	Zeitdifferenz	
A	Beschleunigung	
V	Geschwindigkeit	
T	Zeit	
E	Elongation	nach Anpassung an Koordinatensystem
Z	Zeit	nach Anpassung an Koordinatensystem
L, S, Q, L	Hilfsvariablen	

5. Programmlisting

```
10 !
20 !   UEBUNGSPROGRAMM "SCHWINGUNGEN"
30 !
40 !          04. 12. 1986
50 !
60 !
70 !      AUTOR: DR. H.-J.LOEHR
80 !
90 WINDOW:CLS:PRINT:PRINT:PRINT:PRINT:PRINT
100 PRINTSPC(8)"**********************"
110 PRINTSPC(8)"*                    *"
120 PRINTSPC(8)"*    SCHWINGUNGEN    *"
130 PRINTSPC(8)"*                    *"
140 PRINTSPC(8)"**********************"
150 !
160 PAUSE 60:WINDOW:CLS
170 PRINT:PRINT:PRINT:PRINT"BITTE WERTE EINGEBEN!"
180 PRINT:PRINT:INPUT"FEDERKONSTANTE ......(N/m).... k=";K
190 IF K=0 THEN180
200 PRINT:INPUT"MASSE .................(Kg).... m=";M
210 IF M=0 THEN200
220 C=K/M:FU=1/2/PI*SQR(C):PRINT:PRINT:PRINT
230 PRINT"  -Betrag der Rueckstellkraft F=";K;"N":PRINT
240 PRINT"  -Eigenfrequenz des ungedaempften"
250 PRINT"   Systems fu=";INT(FU*100+.5)/100;"Hz":PRINT:TU=1/FU
260 PRINT:INPUT"REIBUNGSKONSTANTE ...(Kg/s)... r=";B:PRINT
270 PRINT:PRINT:PRINT"ERREGERSCHWINGUNG:  Fa in N , fe in Hz":PRINT:PRINT
280 INPUT"AMPLITUDE DER AEUSSEREN KRAFT  Fa=";FA
290 PRINT:INPUT"FREQUENZ DER AEUSSEREN KRAFT  fe=";FE
300 PAUSE 10:CLS
310 R=B/M:DI=C-R*R/4:IF DI(=0 THEN360
320 OM=SQR(DI):TD=2*PI/OM
330 IF DI()C THEN360:ELSE IF FA()0 AND FE()0 THEN410
340 IF FA=0 OR FE=0 AND B=0 THEN350:ELSE360
350 PRINT:PRINTSPC(2)"FREIE UNGEDAEMPFTE SCHWINGUNG":TD=TU:PRINT:GOTO450
360 IF FA=0 OR FE=0 THEN370:ELSE410
370 PRINT:PRINTSPC(2)"FREIE GEDAEMPFTE SCHWINGUNG"
380 IF DI)0 THEN PRINT:PRINT"SCHWINGFALL";:GOTO450
390 IF DI(0 THEN PRINT:PRINT"KRIECHFALL":TD=TU:GOTO470
400 PRINT:PRINT"APERIODISCHER GRENZFALL":TD=TU:GOTO470
410 PRINT:PRINT"ERZWUNGENE SCHWINGUNG":PRINT:PRINT"ERREGERSCHWINGUNG:"
420 PRINT:PRINT"FREQUENZ fe=";FE;"Hz";
430 PRINTSPC(3)"AMPLITUDE Fa=";FA;"N":GOSUB1030:!*UP KOORDINATENSYSTEM*
440 GOTO510
450 PRINTSPC(2)"f=";INT(1/TD*100+.5)/100;"Hz";
460 PRINTSPC(4)"T=";INT(TD*100+.5)/100;"s"
470 WINDOW 6,31,0,39:CLS:GOSUB1030:!***UP KOORDINATENSYSTEM***
480 !
490 !***BERECHNUNG DER MOMENTANWERTE***
500 !
510 X=1:D=INT(TD+.5)/100
520 X=X+V*D :!***ELONGATION***
530 A=FA/M*SIN(2*PI*FE*T)-C*X-R*V :!***BESCHLEUNIGUNG***
540 V=V+A*D  :!***GESCHWINDIGKEIT***
550 T=T+D    :!***ZEIT***
560 E=INT(68*X)+92 :!***ANPASSUNG ELONGATION AN ORDINATENACHSE***
570 Z=INT(T/2/TD*232)+49-S*232 :!***ANPASSUNG ZEIT AN ABSZISSENACHSE***
580 PSET Z,E,7
590 IF T(=(S+1)*2*TD THEN520:!***DARSTELLUNG VON 2 PERIODEN***
600 S=S+1:!***NAECHSTES PERIODENPAAR***
610 WINDOW 8,10,23,39
620 INPUT"GRAFIK LOESCHEN?   Y/N";L$:IF L$="N" THEN CLS:GOTO520
630 WINDOW 7,31,0,39:CLS:GOSUB1030:!***UP KOORDINATENSYSTEM***
640 GOTO520
650 END
```

```
660 !*****************************
1000 !
1010 ! UP "KOORDINATENSYSTEM"
1020 !
1030 FOR L=48 TO 290
1040 PSET L,92,7
1050 NEXT L
1060 PRINTAT(21,36);"t"
1065 IF S)=1 THEN RETURN
1070 FOR Q=24 TO 170
1080 PSET 48,Q,7
1090 NEXT Q
1100 PRINTAT(10,5);"X"
1110 RETURN
1120 !*****************************
2000 !
2010 ! UP "EINSCHWINGZEIT"
2020 WINDOW 8,10,23,39:CLS
2030 PRINT"T=";INT(T*FE)"*Te"
2040 RETURN
2050 !*****************************
```

6. Hardcopy-Auswahl

```
BITTE WERTE EINGEBEN!

FEDERKONSTANTE .....(N/m).... k= 1
MASSE ..............(Kg).... m= 1

   -Betrag der Rueckstellkraft F= 1 N
   -Eigenfrequenz des ungedaempften
    Systems fu= .16 Hz

REIBUNGSKONSTANTE ...(Kg/s)... r= .05

ERREGERSCHWINGUNG:  Fa in N , fe in Hz

AMPLITUDE DER AEUSSEREN KRAFT  Fa= 2.5
FREQUENZ DER AEUSSEREN KRAFT   fe= .5
```

Bild 1. Bildschirm nach Parametereingabe für das Beispiel in Bild 6

```
FREIE UNGEDAEMPFTE SCHWINGUNG
f= .16 Hz     T= 6.28 s
```

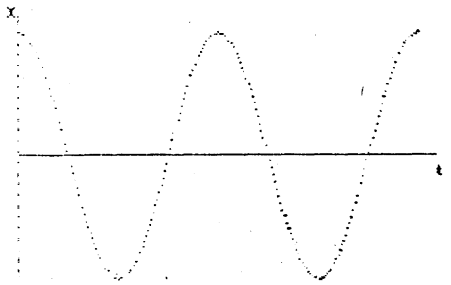


Bild 2. $k = 1$ N/m, $m = 1$ kg, $r = 0$ kg/s, $F_a = 0$ N, $f_e = 0$ Hz

```
FREIE GEDAEMPFTE SCHWINGUNG
APERIODISCHER GRENZFALL
```

Bild 3. $k = 1$ N/m, $m = 1$ kg, $r = 0{,}5$ kg/s, $F_a = 0$ N, $f_e = 0$ Hz

```
FREIE GEDAEMPFTE SCHWINGUNG
SCHWINGFALL   f= .15 Hz
T= 6.49 s
```

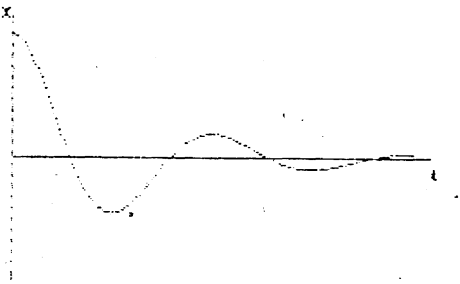


Bild 4. $k = 1$ N/m, $m = 1$ kg, $r = 2{,}0$ kg/s, $F_a = 0$ N, $f_e = 0$ Hz

```
FREIE GEDAEMPFTE SCHWINGUNG
KRIECHFALL
```

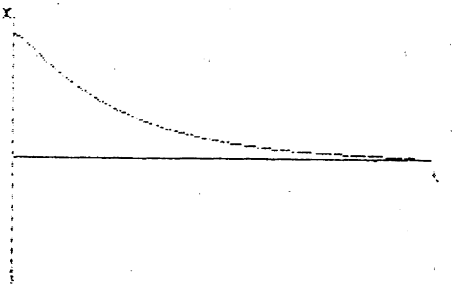


Bild 5. $k = 1$ N/m, $m = 1$ kg, $r = 3{,}5$ kg/s, $F_a = 0$ N, $f_e = 0$ Hz

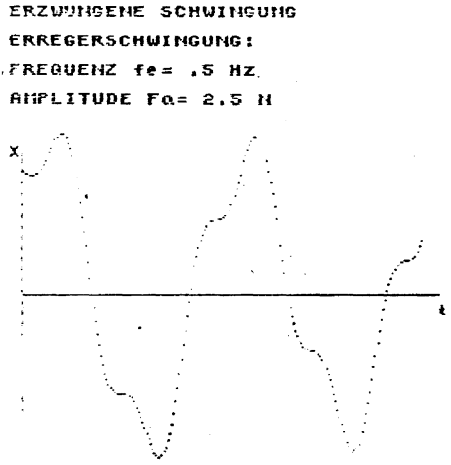


Bild 6. a. $k = 1$ N/m, $m = 1$ kg, $r = 0{,}05$ kg/s, $F_a = 2{,}5$ N, $f_e = 0{,}5$ Hz

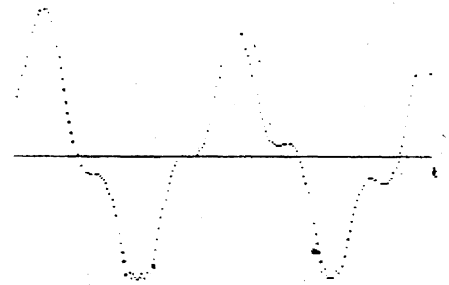


Bild 6b. $k = 1$ N/m, $m = 1$ kg, $r = 0{,}05$ kg/s, $F_a = 2{,}5$ N, $f_e = 0{,}5$ Hz

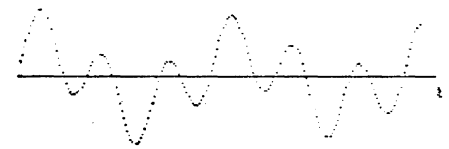


Bild 6c. $k = 1$ N/m, $m = 1$ kg, $r = 0{,}05$ kg/s, $F_a = 2{,}5$ N, $f_e = 0{,}5$ Hz (Perioden 11 und 12, bezogen auf dieSchwingungsdauer der Grundschwingung)

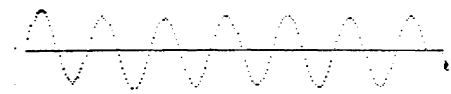


Bild 7b. $k = 1$ N/m, $m = 1$ kg, $r = 0{,}5$ kg/s, $F_a = 2{,}5$ N, $f_e = 0{,}5$ Hz

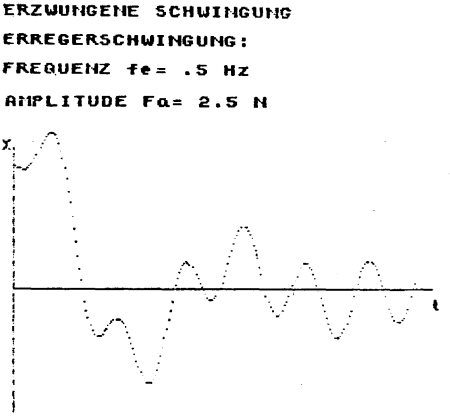


Bild 7a. $k = 1$ N/m, $m = 1$ kg, $r = 0{,}5$ kg/s, $F_a = 2{,}5$ N, $f_e = 0{,}5$ Hz

7. Schlußbemerkungen

In diesem Beitrag wird einerseits eine Möglichkeit zur Untersuchung von Schwingungsvorgängen an Hand von KC-Simulationen vorgestellt. Besondere Aufmerksamkeit sollte dabei den erzwungenen Schwingungen, insbesondere ihren Einschwingvorgängen, gelten. Im Realexperiment erfordern entsprechende Untersuchungen (mit relativ freier Parameterwahl) einen großen gerätetechnischen Aufwand, experimentelle Fertigkeiten sowie Exaktheit, Geduld und viel Zeit bei der Gewinnung aussagekräftiger Meßwertreihen.

Andererseits sollte mit der dem Artikel zugrunde liegenden Gliederung eine Empfehlung zur Form von ausführlichen Programmdokumentationen gegeben werden.

Autor:
OL Dr. Hans-Joachim Löhr
Friedrich-Schiller-Universität Jena
Sektion Physik
WB Methodik des Physik- und Astronomieunterrichts

Simulation auf Mikrorechnern: Spiele und Experimente (Teil 3)

Graphische Ausgabe simulierter Prozesse

In den ersten beiden Teilen dieses Beitrages sind Methoden und Algorithmen zur Erzeugung von Zufallszahlenfolgen vorgestellt worden. Anschließend ist gezeigt worden, wie man mit ihrer Hilfe Prozesse simulieren kann, um Voraussagen über deren Verhalten abzuleiten. Im abschließenden dritten Teil sollen nun einige Anregungen zur graphischen Darstellung simulierter Prozesse vermittelt werden.

1. Zur Geschichte von Simulation und Computergraphik

Während der ersten beiden Jahrzehnte der Entwicklung der Simulationstechnik waren simulierte Prozesse rechnerinterne Vorgänge, die sich der menschlichen Anschauung entzogen. Während eines Simulationslaufes wurden Beobachtungsdaten über den simulierten Prozeß gesammelt, die nach Abschluß dieses Laufes in Form von Einzelwerten, als Mittelwerte, Häufigkeitstabellen oder Zeitreihen, ausgegeben wurden.

Das in den sechziger Jahren entwickelte Simulationssystem GPSS (General Purpose Simulation System) erzeugt für jedes Modell eine Standardausgabe. Sie enthält eine Vielzahl von Werten. Diese Werte beziehen sich entweder auf den beim Abbruch des Simulationslaufes erreichten Zustand des Modells oder sind während dieses Laufes gebildete Summen, Mittelwerte, Standardabweichungen oder Häufigkeiten. Dasselbe gilt für das in der DDR entwickelte, auf ESER-Rechnern verfügbare Simulationssystem SIMDIS [1] oder das auf Basis von FORTRAN entwickelte SIMFOR [2], das auch für PC verfügbar ist. Auf diese Weise entstehen oft umfangreiche, viele Druckseiten oder Bildschirme füllende Datenmengen. Für den Programm- und Modellentwickler sind diese Datenmengen insbesondere beim Programmtest wertvolle Hilfsmittel. Für den Nutzer des Modells, z. B. für den Planungs- und Projektierungsingenieur ist es hingegen sehr schwer, aus diesen Datenmengen eine anschauliche Vorstellung über den Prozeßverlauf abzuleiten und Wesentliches von Unwesentlichem zu trennen.

Seit dem Beginn der siebziger Jahre sind zwei Entwicklungsrichtungen erkennbar, die beide dazu beitragen, den simulierten Prozeß der Dunkelheit rechnerinterner Vorgänge zu entreißen und der Anschauung zugänglich zu machen:

(1) Es werden interaktive Simulationssysteme und Simulationsmodelle geschaffen, die einen Dialog von Modell und Nutzer während des Simulationslaufes gestatten.

(2) Der simulierte Prozeß wird durch Computer veranschaulicht.
Einer der ersten Berichte über die Nutzung der damals noch neuen Bildschirm- und Plottertechnik zur interaktiven Nutzung und graphischen Resultatausgabe von Simulationsmodellen findet man in [3]. In diesem Buch wird das Simulationsmodell eines Flughafens beschrieben. Auf dem Bildschirm wird die Bewegung von Flug- und Fahrzeugen auf dem Flughafengelände trickfilmartig dargestellt.
Damals mußte interaktive, mit graphischer Ausgabe ausgestattete Simulationssoftware für jeden Anwendungsfall einzeln gestaltet werden. Während für die Grundaufgaben der Prozeßsimulation leistungsfähige Software in Gestalt universeller Simulationssysteme bereitstand, die seit Mitte der siebziger Jahre auch zunehmend mit Dialogkomponenten ausgestattet wurde [4], blieb die graphische Resultatdarstellung noch längere Zeit ein Problem, das jeder Anwender selbst zu lösen hatte.
Die Verschiedenartigkeit benutzter Hardware- und Softwaresysteme stand der Entwicklung universeller Software längere Zeit im Wege. Für 16- und 32-Bit-Prozessorsysteme wurde inzwischen universelle Software zur Erzeugung künstlicher, beweglicher Bilder der simulierten Prozesse entwickelt.
Auch für einige Mikrorechner auf Basis der 8-Bit-Prozessortechnik gibt es heute rechnerspezifische Software zur Bildanimation. Portable, zwischen Rechnern verschiedenen Typs übertragbare Lösungen sind hier kaum zu erwarten. Bei der Bildanimation mittels dieser Rechner ist man noch weitgehend, ebenso wie bei der Basissimulationssoftware, auf individuelle Lösungen angewiesen.
An dieser Stelle soll nur erwähnt werden, daß es für ESER-Rechner in Gestalt des Programmiersystems SIMDIS-3 [5] leistungsfähige interaktive Simulationssoftware mit bestimmten Graphikkomponenten für Zeitreihen und Häufigkeitstafeln (Polygonzüge und Säulen) gibt.
Nach dieser Übersicht sollen nun die Mikrorechner wieder zu ihrem Recht kommen.
Während es in den ersten beiden Teilen dieses Beitrages möglich war, eine vom Rechnertyp unabhängige Darstellung zu wählen, muß diese Darstellungsweise in diesem dritten Teil leider aufgegeben werden. Bisher war es möglich, mit BASIC-Sprachelementen auszukommen, die in den meisten BASIC-Versionen verfügbar sind. Da die Computergraphik auf Mikrorechnern noch rechnerspezifisch ist, muß ein Rechnertyp ausgewählt werden. Es liegt nahe, den KC 85/3 zu favorisieren. Seine Verbreitung und seine Vollgraphik sprechen für diese Wahl.

2. Pseudo- und Vollgraphik

Wenn man von der physischen Beschaffenheit eines Bildschirmes abstrahiert, so bleibt sein logisches Grundschema. Logisch ist der Bildschirm ein rechteckiges Rasterfeld. Als Flächenelement in diesem Raster kann

- ein rechteckiges Schema von Punkten als Grundfläche eines Zeichens oder
- ein einzelner Bildpunkt auftreten.

Die kleinsten Teile eines Computerbildes heißen Bildelemente oder Pixel (picture element). Viele Computer nutzen ihren Bildschirm nur in der zuerst geschilderten Weise.
Den Aufbau von Computerbildern aus Bildelementen von der Größe eines Buchstabens bezeichnet man als **Pseudo- oder Quasigrafik**. Der KC 85/3 hat ein festes Raster von 40*32 Zeichenfeldern für Buchstaben, Ziffern oder alle möglichen Sonderzeichen. Jedes dieser Zeichen wird in einem Qua-

drat mit einer Seitenlänge von 8 Bildpunkten dargestellt.
Viele Rechner verfügen über eine zweite Art der Bildschirmnutzung. Den Aufbau von Computerbildern aus einzelnen Punkten bezeichnet man als **Voll-, Punkt- oder Rastergraphik**. Der KC 85/3 verfügt über ein Raster von 320*256 Bildpunkten.
Man hat also die Qual der Wahl zwischen zwei Wegen.
Vorteile der Pseudographik liegen

- im hohen Tempo von Bildaufbau und -modifikation,
- in der wiederholten Nutzung von Bildsymbolen für simulierte Objekte.

Vorteile der Punkt- oder Vollgraphik liegen dagegen in der Darstellungsmöglichkeit individueller Details. Diese Detaildarstellung kostet Zeit, so daß von Fall zu Fall zu entscheiden sein wird, welcher der beiden Wege beschritten oder wie beide Wege kombiniert werden sollen. Die Bilder 1 und 2 enthalten ein BASIC-Programm für einen Zeitvergleich von Punkt- und Pseudographik beim Zeichnen horizontaler Geraden. Dieser Zeitvergleich verdeutlicht, daß man zwischen Präzision und Tempo zu wählen hat.

```
10 ' PUNKTGRAPHIK
20 CLS:C$="10 LINIEN MIT "
30 E$(1)="PSET":E$(2)="LINE"
40 E$(3)="Pseudografik"
45 GOTO110
50 '
60 LOCATE9*I-9,0:PRINTC$+E$(I)
70 INPUT"UHR STARTEN UND ENTER!";X
80 RETURN
90 BEEP: INPUT"GEMESSENE ZEIT:";T(I)
100 RETURN
110 I=1:GOSUB60
120 FOR J=1 TO 10
130     Y=232-5*J
140     FOR X=24 TO 296: PSET X,Y,J
150  NEXT X,J: GOSUB90
160 I=2: GOSUB60
170 FOR J=1 TO 10
180     Y=160-5*J
190     LINE 24,Y,296,Y,J
200 NEXT: GOSUB90
210 I=3:GOSUB60
220 FOR J=1 TO 10
230     PRINTAT(J+20,3);STRING$(34,"_")
240 NEXT:GOSUB90:LOCATE 23,0
250 PRINT"RESULTATVERGLEICH"
255 PRINT"Zeit fuer 10 Linien"
260 FOR I=1 TO 3
270     PRINT"   mit ";E$(I);TAB(25);T(I)
280 NEXT: END
```

Bild 1. BASIC-Programm zum Zeitvergleich von Punkt- und Pseudographik beim Zeichnen horizontaler Linien

```
10 LINIEN MIT PSET
UHR STARTEN UND ENTER!
GEMESSENE ZEIT:_30

10 LINIEN MIT LINE
UHR STARTEN UND ENTER!
GEMESSENE ZEIT:_5

10 LINIEN MIT Pseudografik
UHR STARTEN UND ENTER!
GEMESSENE ZEIT:_1.5

RESULTATVERGLEICH
Zeit fuer 10 Linien
   mit PSET              30
   mit LINE              5
   mit Pseudografik      1.5
OK
>
```

Bild 2. Bildschirmkopie zum Zeitvergleichsprogramm

Auf vielen Mikrorechnern wird der Entwurf eigener Zeichen durch spezielle Programme unterstützt. Die Tastaturbelegung ist auswechselbar und kann um selbst entworfene Sonderzeichen ergänzt werden. Beim Entwurf eigener Zeichen mit Hilfe spezieller Programme werden auf dem Bildschirm vergrößerte Darstellungen entworfen, die dann in Originalgröße gezeigt und nach Wunsch gespeichert werden können. Die Bilder 3 bis 5 stellen Bildschirminhalte während der Entwicklung eines Sonderzeichens vor. Wer nicht über derartige Spezialprogramme verfügt, muß seine Zeichen punktweise konstruieren und in eine hexadezimale Darstellung umsetzen.

3. Parallele und Post-Run-Auswertung

Beim früher dominierenden Stapelbetrieb war der Nutzer meist abwesend, wenn die von ihm geplanten und programmierten Simulationsexperimente liefen, d.h. auf dem Rechner ausge-

```
ZEICHENKONSTRUKTEUR

                AUTOR WALTER GOERGENS

BENUTZUNGSHINWEISE

DIESER ZEICHENKONSTRUKTEUR BEAR-
BEITET DIE ZEICHEN IM BEREICH
DER KLEINEN BUCHSTABEN.

BEI BEDARF SIND DIESE VORHER
MITTELS LOAD EINZULESEN. DIE
AUSGABE HAT AM SCHLUSS MITTELS
SAVE BD00 C000 ZU ERFOLGEN.
IM NUTZERPROGRAMM WERDEN DIE SO
ERSTELLTEN SONDERZEICHEN DURCH
VPOKE 14249,188 AKTIVIERT.
(EINLESEN NICHT VERGESSEN!)
```

Bild 3. Benutzungshinweise eines Programmes zur Konstruktion von Zeichen

```
          BEDIENUNGSANLEITUNG

C   -   KOPIEREN AUS DEM ORGINAL-
        ZEICHENSATZ
N   -   NEUBEGINN
R   -   ZURUECKHOLEN EINES BILDES
Y   -   BILDPUNKT SETZEN
SPC-    (LEER) BILDPUNKT LOESCHEN
        Y,SPC UND KURSORTASTEN
        WIRKEN REPETIEREND
S   -   BILD SPIEGELN
D   -   QUADRATISCHES BILD DREHEN
I   -   BILD INVERTIEREN
G   -   BILD GLEITEN
K   -   BILD KOMPLETIEREN
B   -   BILD BESCHNEIDEN
E   -   BILD ERWEITERN
A   -   ORGINALFORMATIGE ANZEIGE
Q   -   AUSGABE
H   -   BEDIENUNGSHINWEISE

EIN BLINKENDER BILDPUNKT ZEIGT
KURSORPOSITION UND BEREITSCHAFT
ZUM KOMMANDOEMPFANG AN.
```

Bild 4. Menü des Zeichenkonstruktionsprogrammes

führt wurden. Aus der Sicht dieses Nutzers fand jede Resultatauswertung »post run«, also »nach dem Lauf«, der Simulation statt. Heute ist durch die Verbreitung von Personalcomputern und durch Computernutzung im Teilnehmerbetrieb die Anwesenheit des Nutzers während des Simulationslaufes möglich geworden. Damit werden für

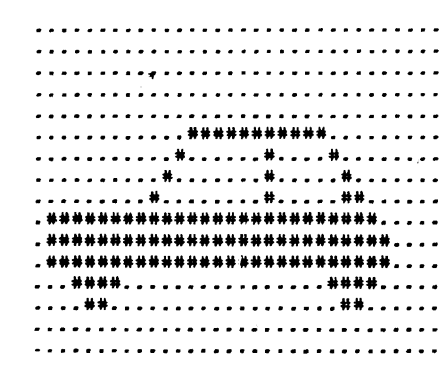


Bild 5. Bildschirmkopie von der Nutzung des Zeichenkonstruktionsprogrammes

den Nutzer eines Simulationsmodells zwei Formen der Resultatdarstellung unterscheidbar:

- die traditionelle Post-Run-Darstellung und
- die während des Simulationslaufes stattfindende Darstellung.

Beide Formen haben ihre Vor- und Nachteile. Die traditionelle **Post-Run-Darstellung** bindet mehr Speicherplatz, weil sie dazu zwingt, während des Laufes mehr Daten zu sammeln und bis zum Schluß aufzubewahren. Mit dieser Datenkonserve kann man dann aber zu beliebigen Zeitpunkten und in wählbarem Tempo arbeiten.

Die **Paralleldarstellung** verarbeitet viele Daten zu ihrem Entstehungszeitpunkt und verzichtet dann auf die Speicherung. So benötigt sie weniger Speicherplatz und unterstützt die interaktive Arbeit mit dem Modell. Aber das Tempo der Datenerzeugung und -bereitstellung zu steuern wird schwieriger. Bei der Simulation komplizierter Prozesse muß der Experimentator oft zu lange warten. Mit der Datenkonserve könnte er Zeitraffung oder Zeitlupe organisieren und seinen Trickfilm beliebig oft betrachten oder auch rückwärts laufen lassen.

Rücksichtnahme auf die oft noch bescheidenen Speicherkapazitäten von

Mikrocomputern führt häufig zur Beschränkung auf die Paralleldarstellung. Wenn der Mikrocomputer allerdings für ein großes Modell zu klein wird, kann er sich bei der Analyse und graphischen Darstellung der vom großen Bruder erzeugten Datenkonserve nützlich machen und die Resultate am Arbeitsplatz zur Auswertung bereitstellen.
Das folgende Programmbeispiel vermittelt einen Eindruck vom Geschwindigkeitsunterschied zwischen Post-Run- und Paralleldarstellung (Bild 6).

```
LIST300

300 !
310 !
320 ! VERGLEICH VOM PARALLEL-
330 ! UND POST-RUN-DARSTELLUNG
340 !
350 ! Z W E I  W U E R F E L
360 !
370 DIM H(12): CLS
380 DEFFNW(X)=INT(6*RND(X)+1)
390 PRINTAT(2,4);"PARALLELDARSTELLUNG"
400 FOR I=2 TO 12
410 PRINTAT(I+24);I:NEXT
420 W2= FNW(1) + FNW(1)
430 H(W2)=H(W2)+1
440 PRINTAT(W2+2,H(W2)+10);"*"
450 IF H(W2)<20 THEN 420
460 PRINTAT(17,4);"POST-RUN-GRAPHIK"
470 FOR I=1 TO 12
480 PRINTAT(I+17,11);STRING$(H(I),"*")
490 NEXT: PAUSE: END
OK
>
```

Bild 6. Programmbeispiel zur Post-Run- und Paralleldarstellung

Zur Konzentration auf das Wesentliche wurde ein besonders einfaches Beispiel gewählt. Beim Spiel mit zwei Würfeln soll die Gesamtzahl der Augen jedes Wurfes erfaßt und durch ein Balkendiagramm dargestellt werden. Dieses Balkendiagramm wird zuerst parallel zum simulierten Würfel und danach noch einmal »post-run« aufgebaut (Bild 7).

4. Graphische Darstellung statistischer Größen

Wenn Simulationsmodelle mit dem Ziel eines Gewinns an Erkenntnissen über

```
OK
>
   PARALLELDARSTELLUNG
    2      **
    3      *******
    4      *********
    5      ******
    6      ****************
    7      ********************
    8      *************
    9      *****
    10     ******
    11     **********
    12     *****

   POST-RUN-GRAPHIK

          **
          *******
          *********
          ******
          ****************
          ********************
          *************
          *****
          ******
          **********
          *****
```

Bild 7. Bildschirmkopie zur Post-Run- und Paralleldarstellung

das simulierte Objekt benutzt werden, so müssen während des Laufes statistische Daten gesammelt werden. Für Einzelwerte, wie Summen, Durchschnitte oder Standardabweichungen, sind graphische Darstellungen selten sinnvoll. Die graphische Darstellung wird dagegen zum fast unverzichtbaren Hilfsmittel der Resultatauswertung, wenn größere Datenmengen auszuwerten sind. Solche Datenmengen sind in der Regel Häufigkeitstafeln oder Zeitreihen. Am Ende des vorigen Abschnittes ist an einem Programmbeispiel gezeigt worden, wie leicht man parallel zum Simulationslauf oder danach ein **Histogramm**, also eine graphische Darstellung einer Häufigkeitstafel, erzeugen kann. Neben der Darstellungsform des Säulen- oder Balkendiagramms gibt es mehrere andere Möglichkeiten. Hier sei nur die sogenannte Tortengraphik genannt, die unterschiedliche Mengenanteile durch verschieden große Torten-

```
290 !
300 ! BESTIMMUNG DER PARAMETER
310 !
320 PRINT,"GRUENPAHSE"
330 INPUT "Laenge in sec";LG
340 INPUT "Zeitlicher Abstand";ZG
350 AU=INT(LG/10)
360 INPUT "Fahrzeuge pro Minute";FA
370 INPUT"Abbruchzeitpunkt:";AB
390 MI=60/FA: OG=2*MI
400 REM INITIALISIERUNG
410 TA=INT(RND(-1)*OG):W=0:WA=0:TG=ZG
420 ZI=0
430 IF VN=1 THEN GOSUB 1000:ELSE GOSUB       2000
440 REM ENTSCHEIDUNG
450 IF TA>=TG THEN GOTO510
460 ZI=ZI+(TA-ZE)*W
470 IF TA=0 THEN ML=0:ELSE ML=ZI/TA
480 W=W+1: ZE=TA: TA=INT(TA+RND(1)*OG)
490 ON VN GOSUB 1500,3000,5000,6000,         7000,6000
500 GOTO 570
510 REM TA>=TG
520 ZI=ZI+(TG-ZE)*W
530 IF TG=0 THEN ML=0:ELSE ML=ZI/TG
540 W=W-AU: IF W<0 THEN W=0
550 ZE=TG: TG=INT(TG+ZG)
560 ON VN GOSUB 1500,4000,5000,4000,         4000,8000
570 REM ENDE DER ZEIT ERREICHT
580 IF TA<AB OR TG<AB THEN GOTO 450
590 ZE=AB
600 IF VN>1 THEN GOTO630
610 GOSUB 1500: PAUSE : GOTO 10
630 ZI=ZI+(AB-ZE)*W
640 IF AB=0 THEN ML=0:ELSE ML=ZI/AB
650 PRINT AT(25,21);W
660 PRINT AT(26,21);INT(ML*100)/100
670 PRINT AT(5,18);AB
680 PAUSE: GOTO 10
```

Bild 8. Programmzeilen für das Simulationsmodell

stücke darstellt. Wer Platz auf dem Bildschirm sparen will, sollte kleine Bildschirmfenster für die Graphik nutzen und darin Histogramme mittels Voll- oder Punktgraphik erzeugen. Wenn man für jeden Balken nur eine Linie und für jeden neuen Wert nur einen Punkt benutzt, ist diese Punktgraphik nicht langsamer als die Pseudographik.

Die zweite häufig auftretende Resultatform ist die **Zeitreihe**. Sie läßt sich als Diagramm in einem Koordinatensystem abbilden, dessen eine Achse die Zeit darstellt. Als Demonstrationsbeispiel wird im folgenden eine Straßeneinmündung benutzt. Die Fahrzeuge kommen in zufälligen Zeitabständen an dieser Einmündung an. Eine Ampel schaltet in einem Abstand ZG auf Grün. Die Länge der Grünphase ergibt sich aus dem Parameter LG. Ein Fahrzeug benötigt zum Verlassen der Nebenstraße 10s. Das Bild 8 zeigt ein Simulationsprogramm zur Nachbildung des Verkehrstaus an dieser Einmündung. Innerhalb dieses Programms kann zwischen verschiedenen Varianten VN der graphischen Darstellung gewählt werden. In der Abhängigkeit von der Variantennummer wird das entsprechende Unterprogramm aufgerufen.

In Bild 9 ist das Anfangsmenü zur Variantenauswahl aufgezeigt. Die einzelnen Varianten werden in den folgenden Abschnitten besprochen und Bild 10 zeigt eine von diesem Programm er-

```
VARIANTENAUSWAHL

A Grafische Darstellung          1
B Laengendarstellung
B 1. Stringzeiger                2
B 2. Stringaufbau                3
C Bewegung ankommend. Objekte
C 1. Kopf- und Endezeiger        4
C 2. Lesen im Bildwiederhol-
     speicher                    5
D Bewegung der Schlange          6

Auswahl :                        2
```

Bild 9. Bildschirmkopie des Anfangsmenüs zur Variantenauswahl

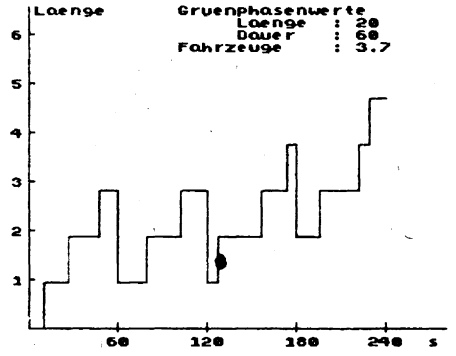


Bild 10. Bildschirmkopie der graphischen Zeitreihendarstellung

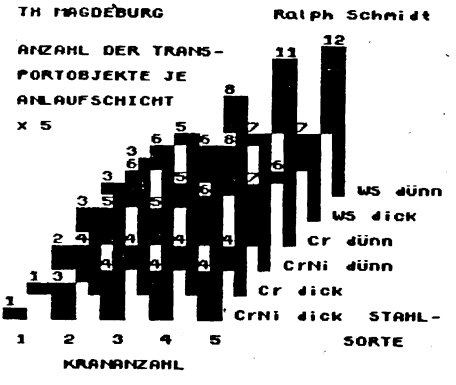


Bild 11. Beispiel einer dreidimensionalen Ergebnisdarstellung

zeugte graphische Zeitreihendarstellung der Warteschlangenlänge.
Resultatsgrößen sind mehr oder weniger stark von Modell- oder Eingabeparametern abhängig. Zur Darstellung dieser Abhängigkeit sind oft dreidimensionale Darstellungen erwünscht. In Bild 11 wird ein Beispiel zur dreidimensionalen Darstellung gezeigt, das auf dem Bildschirm farbig gestaltet ist und auf diese Weise die Abhängigkeit eines Resultatsparameters von zwei Modellparametern veranschaulicht.

5. Bildanimation

Die im vorigen Abschnitt behandelten Formen der Resultatdarstellung lassen sich relativ leicht programmieren und auch für andere Problemstellungen nutzen. Ein neues Anspruchsniveau an den Programmierer erwächst aus der Forderung nach beweglichen Prozeßbildern. Bildanimation bedeutet die künstliche Erzeugung beweglicher Bilder, die dem Original nicht abstrakt, sondern optisch ähnlich sind.
Auf dem Gebiet der Bildanimation sind seit dem Aufkommen der Telespiele zu Beginn der siebziger Jahre breite und schnelle Fortschritte gemacht worden. Computerspiele erreichen heute eine in Anbetracht der begrenzten Möglichkeiten der 8-Bit-Prozessortechnik beeindruckende Realitätstreue in Bild und Ton. In ihnen verkörpern sich Fleiß und Erfahrung talentierter Programmierer, die oft in einer Gruppe von 4 bis 6 Personen mehr als ein Jahr mit der Produktion eines guten Spiels beschäftigt sind.
Die meisten Computerspiele erzeugen eine künstliche Welt, eine Spielzeugwelt, in der Prozesse teilweise unabhängig vom Willen des Spielers, teilweise aber von ihm gesteuert ablaufen und auf dem Bildschirm verfolgt werden können. Spiele unterscheiden sich nur

in ihrer Zielstellung, nicht aber in ihren Werkzeugen und Darstellungsmethoden von Simulationsanwendungen mit den Zielen des Trainings oder des Erkenntnisgewinns. Computerspiele haben eine massenhafte Verbreitung gefunden. In ihre Entwicklung ist wesentlich mehr Zeit und Geld investiert worden als in die Entwicklung anderer Simulationsanwendungen auf Mikrorechnern. Heute kann jeder, der Mikrorechner zur Simulation und Bildanimation nutzen will, musterhaft gestaltete Vorbilder in Form von Computerspielen finden.

5.1. Elementaroperationen zur Bildanimation

Wenn man auf dem Bildschirm ein Stück künstlicher Realität zu gestalten hat, sollte man zuerst überlegen, wie sich die Objekte dieser Scheinwelt klassifizieren lassen. Man sollte versuchen, zwischen

- einem längere Zeit festen, nur im Ganzen auszutauschenden oder zu bewegenden Bildhintergrund und
- Klassen veränderlicher Objekte

zu unterscheiden.

Zur Gestaltung des Hintergrundes eignen sich Mal- oder Zeichenprogramme, die für den KC 85/3 verfügbar sind.

Die Veränderung eines Objektes kann in seiner Orts-, Gestalts- oder Farbänderung, aber auch in seiner Erzeugung oder Vernichtung bestehen.

Nun kann man damit beginnen, Daten zur Objektklassenbeschreibung und Algorithmen zur Objektveränderung zu entwerfen. Spätestens dann braucht man ein Szenarium, das die darzustellenden Prozesse beschreibt. Dieses Szenarium kann durch das Simulationsmodell geliefert werden. Im folgenden sollen Elementaroperationen zur Ortsveränderung von Objekten dargestellt werden.

Die einfachste, nicht noch weiter zerlegbare Operation zur Bildanimation ist die Bewegung eines Elementarobjektes von einer Bildposition zu einer benachbarten Position. Elementarobjekte können dabei ein Punkt oder ein Zeichen sein. Hierbei treten die im 2. Abschnitt erörterten Unterschiede zwischen Punkt- und Pseudographik wieder hervor. Für die Anweisungsfolge zur Punktbewegung

```
PSET X, Y, 1: PSET X+1, Y:
PRESET X, Y
```

wird annähernd ebensoviel Zeit verbraucht wie für die Bewegung eines Zeichens durch

```
?AT(IZ, IS);A$: ?AT (IZ, IS+1);A$:
?AT (IZ, IS); " "
```

Auf ähnliche Weise kann man ein Elementarobjekt auf jeder (x, y)- oder (IZ, IS)-Bahn über den Bildschirm bewegen. Probleme ergeben sich, wenn der Bildschirm nicht leer, sondern von weiteren festen oder beweglichen Objekten bedeckt ist. Im Falle der Kollision zweier Objekte gibt es zwei Möglichkeiten:

(1) ein Objekt liegt im Vordergrund und überdeckt das andere oder
(2) ein Objekt verändert das andere.

Vor einer dieser Reaktionen auf eine Kollision muß diese allerdings erkannt werden.

Die Erkennung von Kollisionen kann im Programm dann leicht realisiert werden, wenn die benutzte Programmiersprache geeignete Anweisungen besitzt. Der BASIC-Interpreter des KC85/3 hat die Funktionen PTEST und VGET$. PTEST zeigt durch ihre Werte (0/1) an, ob eine Bildschirmposition im Punkt- oder Pixelraster frei oder durch einen Punkt besetzt ist. Die Funktion VGET$ hat als Wert das an der aktuellen Cursorposition befindliche Zeichen. Die Bewegung eines Vordergrundzeichens, das vorhande-

nen Bildschirminhalt verdeckt, kann mit Hilfe folgender Anweisungen programmiert werden:

```
1000 INPUT"ZEILENNUMMER?";IZ
1010 WINDOW: A$="@"
1015 LOCATE IZ,5: B$=VGET$
1020 FOR IS=5 TO 30
1030 LOCATE IZ,IS+1: C$=VGET$
1040 IFC$=CHR$(0)THENC$=CHR$(32)
1050 LOCATE IZ,IS:PRINTB$;A$
1060 B$=C$
1070 NEXT:END
```

Eine Bewegung von links nach rechts wird auf ähnliche Weise erzeugt. Vertikale Bewegungen erreicht man nach demselben Algorithmus, wenn man für diese Bewegung ein einspaltiges Bildschirmfenster einrichtet.
Wer mit einem BASIC-Interpreter ohne PTEST und VGET$ arbeitet, sollte entsprechende Unterprogramme im Maschinencode schreiben. Das ist oft besser als eine zum Bildwiederholspeicher parallele Aufzeichnung des Bildschirminhaltes in einem Feld.
Im folgenden Abschnitt wird am Beispiel der Straßeneinmündung die Erzeugung beweglicher Bilder demonstriert.

5.2. Beispiele für die Simulation mit Bildanimation

Als nachzubildender Prozeß wird das im 4. Abschnitt beschriebene Verhalten von Fahrzeugen an einer Straßeneinmündung verwendet. Die aufgeführten Beispiele beziehen sich auf die Darstellung der Fahrzeuge in der Nebenstraße. Es werden die Elementaroperationen zur Bildanimation genutzt. Eine Erweiterung dieser angegebenen Befehlsfolgen wurde vorgenommen, da die Fahrzeuge aus drei Sonderzeichen bestehen. Die Generierung dieser Zeichen erfolgte mit dem in Bild 3 gezeigten Spezialprogramm.
Die graphische Darstellung des Prozesses setzt sich aus zwei Teilen zusammen:

- Erzeugung des permanenten, statischen Bildteiles (Standbild),
- Erzeugung und Bewegung temporärer, dynamischer Objekte (Bildanimation).

Bild 12 zeigt das Standbild für die Straßeneinmündung.

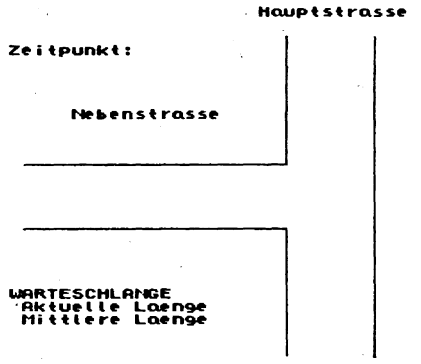


Bild 12. Bildschirmkopie des Standbildes für die Straßeneinmündung

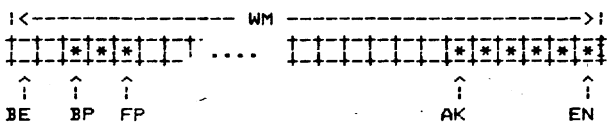
Welche Aufgaben hat die Bildanimation für den vorliegenden Fall zu erfüllen?

1. Ankommende Fahrzeuge müssen in die Warteschlange eingereiht werden. Da der Warteraum nur in einer bestimmten Größe auf dem Bildschirm abgebildet werden kann, muß bei der Lösung dieser Aufgabe berücksichtigt werden, daß ein Überschreiten der darstellbaren Warteschlangenlänge nicht möglich ist.

2. Abfahrende Fahrzeuge sind aus der Warteschlange zu entfernen. Im Simulationsmodell wird nicht die Anzahl der tatsächlich abgefahrenen Fahrzeuge berechnet, sondern die aktuelle Warteschlangenlänge nach der Abfahrt der Fahrzeuge.

Zur Lösung dieser Aufgaben werden zusätzliche Variablen verwendet.
Der Zusammenhang zwischen den Größen AK, BE, BP, EN und FP ist aus der folgenden Darstellung ersichtlich.

Name	Bedeutung	Berechnung
AK	aktuelle Position des letzten Objektes in der Warteschlange	
BE	Beginnposition der darstellbaren Warteschlangenlänge	
BP	aktuelle Endeposition eines Objektes in der Bewegung	
EN	Endeposition der darstellbaren Warteschlangenlänge	EN = WM * OL + BE — 1
FP	aktuelle Anfangsposition eines Objektes in der Bewegung	
LE$	leeres Objekt	LE$=STRING$ (OL, " ")
OB$	Charakterisierung des Objektes	Verkettung der definierten Sonderzeichen
OL	Länge des Objektes in Zeichen	OL=LEN (OB$)
W	aktuelle Warteschlangenlänge	
WM	maximal darstellbare Warteschlangenlänge in Objekten	
ZN	aktuelle Zeilennummer, in der die Darstellung erfolgt	




```
2000 !
2010 ! **** RAHMEN FUER ANIMATION ***
2020 !
2030 ! DARSTELLUNG DER KREUZUNG
2040 WINDOW: CLS:XU=50:YU=20:F1=7
2050 XA=XU:XE=21*8+XA:YA=10*8+YU:YE=YA
2060 DI=INT((XE-XA)/8)
2070 LINE XA,YA,XE,YE,F1
2080 YA=YA+5*8:YE=YA
2090 LINE XA,YA,XE,YE
2100 XA=XE:YE=YA+10*8
2110 LINE XA,YA,XE,YE
2120 YA=YA-5*8:YE=YU
2130 LINE XA,YA,XE,YE
2140 XA=XA+7*8:XE=XA:YA=YU:YE=YA+25*8
2150 LINE XA,YA,XE,YE
2160 PRINT AT(2,25);"Hauptstrasse"
2170 PRINT AT(5,5);"Zeitpunkt:"
2180 PRINT AT(10,10);"Nebenstrasse"
2190 PRINT AT(24,5);"WARTESCHLANGE"
2200 PRINT AT(25,6);"Aktuelle Laenge"
2210 PRINT AT(26,6);"Mittlere Laenge"
2220 ! INITIALISIERUNG VON VARIABLEN
2222 VPOKE 14249,188
2230 OB$=CHR$(123)+CHR$(124)+CHR$(125)
2240 OL=LEN(OB$): WM=INT(DI/OL)
2245 Z1$=LEFT$(OB$,1)
2250 BE=INT(XU/8):EN=WM*OL+BE-1:AK=EN+1
2260 ZN=17:LE$=STRING$(OL," ")
2270 RETURN
```

Bild 13. Programmzeilen zum Aufbau des Standbildes

Die Initialisierung dieser Variablen erfolgt beim Aufbau des Standbildes. Bild 13 enthält die erforderlichen Anweisungen.

Im folgenden werden Varianten zur Lösung von Aufgaben der Bildanimation vorgestellt. Die einzelnen Varianten unterscheiden sich zum einen im Grad der Abbildungsgenauigkeit des Prozesses und zum anderen in ihrer programmtechnischen Implementation. Der Grad der Differenzierung ist durch die Buchstaben A, B, C und D gekennzeichnet. Die Unterschiede in der Implementation sind durch Ziffern gegeben. Das verwendete Simulationsprogramm beinhaltet ein Menü, aus dem die jeweiligen Varianten ausgewählt werden.

Variante A hat die in 4. beschriebene Zeitreihendarstellung zum Inhalt.

In **Variante B** erfolgt nur eine Darstellung der aktuellen Warteschlangenlänge.

B1 Stringzeiger

Für jede Teilaufgabe der Bildanimation wird ein Unterprogramm verwendet. Bei der Darstellung der ankommenden Objekte wird die Position AK neu berechnet, und auf dieser erfolgt das Drucken der Zeichenkette OB$. Eine ähnliche Vorgehensweise ergibt sich bei der Darstellung des Abfahrens der Objekte. Auf die aktuelle Position AK wird die entsprechende leere Zeichenkette LE$ gedruckt und die aktuelle Position neu berechnet. Die beiden Unterprogramme sind in Bild 14 wiedergegeben.

```
3000 !
3010 ! **** STRINGZEIGER - AUFBAU ***
3020 !
3030 PRINT AT(5,18);ZE
3040 PRINT AT(25,21);W
3050 PRINT AT(26,21);INT(ML*100)/100
3060 IF WM-W <0 THEN PAUSE 10: RETURN
3070 AK=AK-OL:PRINT AT(ZN,AK);OB$
3080 PAUSE 10: RETURN
3090 !
4000 !
4010 ! **** STRINGZEIGER - ENTFERNEN *
4020 !
4030 PRINT AT(5,18);ZE
4040 PRINT AT(25,21);W
4050 PRINT AT(26,21);INT(ML*100)/100
4060 IF W>=WM THEN PAUSE 10: RETURN
4065 NK=EN - W*OL + 1
4070 PRINT AT(ZN,AK);LE$: AK=AK+OL
4080 IF AK=NK THEN PAUSE 10: RETURN
4090 PAUSE 10: GOTO 4070
4110 !
```

Bild 14. Programmzeilen zur Variante Stringzeiger

Eine Kopie eines während der Simulation erzeugtes Animationsbildes ist in Bild 15 zu sehen.

B2 Stringaufbau

In der Variante B1 wurde jede Aufgabe der Animation mit einem speziellen Unterprogramm gelöst. Es ist jedoch auch möglich, zur Animation nur ein

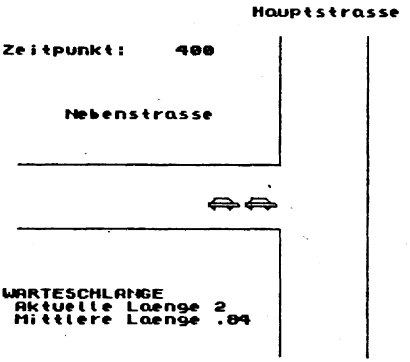


Bild 15. Bildschirmkopie eines erzeugten Animationsbildes

Unterprogramm zu verwenden. Hierzu wird eine Zeichenkette BI$ aufgebaut, deren Länge durch die darstellbare Warteschlangenlänge gegeben ist. Bei jeder Veränderung der Warteschlangenlänge durch das Simulationsprogramm erfolgt der Neuaufbau sowie das anschließende Drucken der Zeichenkette BI$. Die Programmzeilen für diese Variante sind Bild 16 zu entnehmen.

```
5000 !
5010 ! **** STINGAUFBAU ****
5020 !
5030 PRINT AT(5,18);ZE
5040 PRINT AT(25,21);W
5050 PRINT AT(26,21);INT(ML*100)/100
5060 DI=WM-W
5065 IF DI<0 THEN PAUSE 10: RETURN
5070 BI$=STRING$(DI,LE$)+STRING$(W,OB$)
5080 PRINT AT(ZN,BE);BI$
5090 PAUSE 10:  RETURN
5100 !
```

Bild 16. Programmzeilen zur Variante Stringaufbau

Für beide Varianten wurde die benötigte Rechenzeit zur Simulation bei sonst gleichbleibenden Parametern gemessen. Unter Nutzung der Variante B1 betrug die Rechenzeit 2 min und 15 s, während die Rechenzeit bei Verwendung von Variante B2 auf 2 min und 56 s anstieg.
Betrachtet man die mit den Program-

men aus Variante B erzeugten Animationen, so zeigen sie noch wenig Bewegung. Mit den Programmen der **Variante C** wird der Detaillierungsgrad der Abbildung bei der Animation erhöht. Dem Heranfahren der Fahrzeuge an die Kreuzung liegt folgende Idee zugrunde:

Die Bewegung eines Objektes nach rechts wird nachgebildet, indem auf die alte Position die Zeichenkettenkombination " " und OB$ ausgegeben wird.

Für diese Vorgehensweise werden zwei verschiedene Implementationen vorgestellt.

C1 Stringkopf- und Stringendezeiger

Diese Möglichkeit verwendet die Variablen FP und BP zur Kennzeichnung der aktuellen Position des zu bewegenden Objektes. FP bezeichnet die augenblickliche Anfangsposition, während BP die aktuelle Endposition kennzeichnet. Durch einen Vergleich von FP mit AK kann somit ein Auffahren vermieden werden. Der Programmtext ist aus Bild 17 ersichtlich.

```
6000 !
6010 ! **** KOPF- UND ENDEZEIGER ****
6020 !
6030 PRINT AT(5,18);ZE
6040 PRINT AT(25,21);W
6050 PRINT AT(26,21);INT(ML*100)/100
6060 IF W>WM THEN PAUSE 10: RETURN
6070 PRINT AT(ZN,BE);OB$
6080 FP=BE+OL-1: BP=BE
6090 IF FP+1=AK THEN AK=AK-OL: GOTO6115
6100 PRINT AT(ZN,BP);" ";OB$
6110 BP=BP+1: FP=FP+1: GOTO6090
6115 PAUSE 10: RETURN
6120 !
```

Bild 17. Programmzeilen zur Variante Stringkopf- und Stringendezeiger

C2 Lesen im Bildwiederholspeicher

Mit Nutzung der BASIC-Funktionen VGET$ und LOCATE des KC85/3 ergibt sich eine andere Möglichkeit der Implementation. Hierbei prüft das Objekt selbständig, ob rechts neben ihm noch Platz zum Weiterfahren ist. Die erforderlichen Informationen werden aus dem Bildwiederholspeicher gelesen. Bild 18 enthält die notwendigen Programmzeilen.

```
7000 !
7010 ! **** BILDWIEDERHOLSPEICHER ****
7020 !
7030 PRINTAT(5,18);ZE
7040 PRINT AT(25,21);W
7050 PRINT AT(26,21);INT(ML*100)/100
7060 IF W>WM THEN PAUSE 1: RETURN
7070 PRINT AT(ZN,BE);OB$
7080 LOCATE ZN-1,BE+OL
7090 NF$=VGET$: SP=BE
7100 FOR I=1 TO (WM-1)*OL
7110 IF NF$=Z1$ THEN I=1000: GOTO7150
7120 PRINT AT(ZN,SP);" ";OB$
7130 LOCATE ZN-1,SP+OL+1
7140 NF$=VGET$: SP=SP+1
7150 NEXT I:AK=AK-OL
7160 PAUSE 10: RETURN
7170 !
```

Bild 18. Programmzeilen zur Variante Lesen im Bildwiederholspeicher

Variante D beinhaltet das Entfernen der Fahrzeuge aus der Warteschlange. Zur Lösung dieser Aufgabe können verschiedene Möglichkeiten genutzt werden. Die in Bild 19 gezeigte Implementation realisiert folgende Grundidee: Alle Pseudozeichen der Objekte werden gleichmäßig um eine Position nach rechts verschoben. Dadurch entsteht der Eindruck, daß sich die Warteschlange bewegt.

```
8000 !
8010 ! **** BEWEGUNG DER SCHLANGE ****
8020 !
8030 PRINT AT(5,18);ZE
8040 PRINT AT(25,21);W
8050 PRINT AT(26,21);INT(ML*100)/100
8060 IF W>= WM THEN PAUSE 10: RETURN
8070 IF W=WM-1 THEN J=1: ELSE J=AU
8080 FOR I=1 TO J*OL
8090 IF AK>EN THEN I=1000: GOTO8170
8100 SP=EN
8110 LOCATE ZN-1,SP-1: VG$=VGET$
8120 LOCATE ZN-1,SP: PRINT VG$
8130 SP=SP-1
8140 IF SP>AK THEN GOTO8110
8150 LOCATE ZN-1,SP: PRINT " "
8160 AK=AK+1
8170 NEXT I
8180 PAUSE 10: RETURN
8190 !
```

Bild 19. Programmzeilen zur Variante Bewegung der Schlange

Die Programmierung erfolgte unter Verwendung von VGET$ und LOCATE.
Darauf aufbauend kann z.B. ein »Entfernen« programmiert werden, in dem sich die Objekte nacheinander bewegen, wenn sich vor ihnen ein genügend großer Sicherheitsabstand gebildet hat.

5.3. Tempo- und Qualitätssteigerung

Entsprechend dem obigen Beispiel gestaltete Bildanimation hat im Vergleich zu guten Computerspielen auffallende Mängel:

(1) die Bewegungen sind zu sprunghaft und diskontinuierlich, und

(2) die Bildveränderungen erfolgen oft zu langsam.

Die **Kontinuität** der Bewegungen läßt sich verbessern, wenn man Objektbilder um Bruchteile (1/2, 1/4 oder 1/8) einer Zeichenposition verschiebt. Dazu muß man zusätzliche Zeichen bereithalten.
Das **Bewegungstempo** von Objekten, das um so langsamer wird, je mehr Objekte zu bewegen sind, läßt sich vervielfachen, wenn man von BASIC zu einer anderen Programmiersprache übergeht. Hierfür bieten sich PASCAL, FORTH oder die Assemblersprache an. Auch bei Nutzung dieser Hilfsmittel gelangt ein 8-Bit-Prozessor schnell an seine Leistungsgrenzen, wenn

- zeitlich parallel viele Objekte zu bewegen sind,
- die zu bewegenden Objekte aus mehreren Komponenten (Punkten oder Zeichen) bestehen,
- diese Objekte sich bei Kollisionen oder während ihrer sonstigen Bewegung in ihrer Form verändern und
- die Bewegung möglichst kontinuierlich verlaufen soll.

Zur Überwindung solcher Grenzen werden Mikrocomputer heute oft mit zusätzlichen **Graphikprozessoren** (VIC = Video Interface Controller) ausgestattet. Das Zusammenwirken des Zentralprozessors (CPU) mit dem VIC kann dann z.B. in folgender Weise organisiert werden: Die CPU erzeugt im gemeinsam benutzten RAM Muster der beweglichen Objekte, die Movable Object Block (MOB) oder Sprite genannt werden. Der VIC arbeitet mit diesen MOBs. Von der CPU übernimmt er eine Anfangsposition, eine Information über Bewegungsrichtung, -geschwindigkeit und -ende sowie Informationen über Koordinaten, wo Maßstabsänderungen eintreten sollen. Die MOBs tragen Informationen über Prioritäten im Falle der Kollision. Mit diesen Informationen realisiert der VIC selbständig Bewegungen und maßstäbliche Transformationen. Er meldet der CPU Kollisionen oder das planmäßige Ende einer Bewegung. Die CPU verarbeitet diese im RAM gespeicherten Informationen und errechnet neue Steuerinformationen für den VIC. Mit diesen Methoden erreicht man die oftmals erstaunlich hohen Darstellungsqualitäten von Computerspielen. Durch Priorisierung von MOBs und Hintergrund sowie durch schnelle Maßstabstransformation wird der Eindruck einer Bewegung im Raum vermittelt, ohne daß der Computer über ein internes räumliches Modell seiner simulierten Welt verfügt.
Solche internen räumlichen oder Volumenmodelle, die beliebige geometrische Transformationen erlauben, werden wegen ihres hohen Speicherplatz- und auch Rechenzeitbedarfs in der Regel nur auf 32-Bit-Rechnern oder von Multiprozessorsystemen verarbeitet. Ihre Betrachtung liegt außerhalb der hier dem Rahmen der »Kleinstrechner-TIPS« angemessenen Grenzen.

Fortsetzung auf Seite 63

Bewegte Bilder aus dem KC

1. Einleitung

Die hochauflösende Grafik der Kleincomputer ist sicher sowohl eine ihrer interessantesten Eigenschaften als auch eines ihrer weitreichenden Anwendungsgebiete. Sie ermöglicht die Darstellung komplizierter geometrischer Strukturen (»CAD«), aber auch fotografieähnlicher Bilder. Außerdem kann durch den schnellen Wechsel mehrerer Bilder eine Bewegung – wie beim Trickfilm – erzeugt werden. Dazu existieren verschiedene Möglichkeiten, die ich im folgenden kurz vorstellen möchte.

Die einfachste Methode verwendet die BASIC-Befehle PSET und PRESET, womit aber durch die geringe Geschwindigkeit nur minimale Veränderungen von Bild zu Bild vorgenommen werden können.

Eine weitere recht einfache Variante ist das Umdefinieren des Zeichensatzes – aus Buchstaben werden kleine Männchen – und die Ausgabe durch PRINT. Diese Methode wird von den meisten Reaktionsspielen benutzt, ist aber ebenso wie PSET sehr langsam und ermöglicht im allgemeinen nur eine ziemlich ruckartige Bewegung.

Die folgerichtige Weiterentwicklung führt zu den sogenannten »Sprites«. Diese »Kobolde« entsprechen Grafikzeichen, die größer als eine normale PRINT-Position sind (mindestens 16×16 Punkte) und die sich nicht nur pixelgenau auf dem Bildschirm positionieren und bewegen lassen, sondern die noch eine zusätzliche innere Bewegung (Schritte, Armschlenkern) erlauben. Leider ist mir für die KC 85/2, /3, /4 keine Implementierung dieses sehr vorteilhaften Systems bekannt. Es wäre sicher ein lohnendes Thema für eine separate Veröffentlichung.

Großflächige Bewegungen anstelle von kleinen zuckenden Männchen ermöglicht die Methode des Umspeicherns des Bildwiederholspeichers IRM in den RAM und das spätere schnell aufeinanderfolgende Zurückladen mehrerer solcher Bilder. Die Zeitdauer bei der Erzeugung dieser Bilder spielt dabei nur eine untergeordnete Rolle. Sie kann also auch in BASIC erfolgen. Würden wir eine den ganzen Bildschirm erfassende Bewegung auf diese Weise darstellen wollen, erfordert das 10K je Bild, also 10×64K RAM für eine gute, schnelle Bewegung (6 s, 10 Hz). Wegen des großen Speicherplatzbedarfes ist das praktisch nicht realisierbar. Wenn wir uns allerdings mit einer Bewegung innerhalb eines Bildschirmausschnitts (Window) begnügen, kommen wir mit einem 64K RAM Steckmodul aus, wenngleich das Umspeichern der Windows durch den komplizierten Bildwiederholspeicheraufbau etwas schwierig ist. Die Windows müßten dann ana-

log zu den vollständigen Bildschirmen schnell wieder zurückgeladen werden. Je nach vorhandenem RAM kann man dann das verwendete Bildschirmfenster unterschiedlich groß wählen.
Das Ziel dieses Beitrages ist es, mein Programm MOVIE vorzustellen, das nach letzterer Methode arbeitet. Seine Leistungsfähigkeit wird am Beispiel eines drehenden Erdglobus demonstriert.

2. Programmaufbau

Das Programm MOVIE wurde in Maschinensprache mit Hilfe eines Assemblers geschrieben. Ich habe zur Entwicklung nicht das weitverbreitete EDAS, sondern den Assembler GENS verwendet. Der einzige Unterschied besteht beim Textformat in der Darstellung der hexadezimalen Zahlen, die bei GENS mit # eingeleitet werden.
Das Programm MOVIE setzt sich aus folgenden Hauptroutinen zusammen:

1. Initialisierung (INIT)
INIT muß zu Beginn der Arbeit mit dem Programm aufgerufen werden. Es erzeugt wichtige Hilfsvariablen und eine Adreßtabelle.

2. Screen → Memory (SCME)
Diese Routine verlagert ein spezifiziertes Window vom IRM in den RAM.

3. Bildfolge (FOLGE)
FOLGE ist dasjenige Teilprogramm, das die einzelnen (mit SCME abgespeicherten Windows) nacheinander wieder in den IRM zurücklädt.

Die Bilder befinden sich immer in dem Bereich #4000 bis #7FFF. Diese 16K Byte bilden einen »Block«, auf den bei Verwendung eines 64-K-RAM-Moduls (M011) ohne Schwierigkeiten zugegriffen werden kann. Mit Hilfe von Modulumschaltungen (SWITCH) können nacheinander alle 4 Blöcke eines 64-K-Moduls in den Bereich ab #4000 gebracht werden. Beim Abspeichern mit SCME muß der Nutzer selbst dafür sorgen. Beim Vorführen der Bilder übernimmt dies jedoch das Programm.

3. Variablen

Das Programm ist mit einer Vielzahl von Variablen ausgestattet, so daß es für die unterschiedlichsten Zwecke verwendet werden kann.
Für den Nutzer sind folgende Variablen von Bedeutung:

1. YA, YE, XA, XE
Damit werden die Begrenzungen des verwendeten Bildschirmfensters in demselben Format wie bei dem BASIC-Befehl WINDOW angegeben.

2. W1, W2
Um die Bildwechselfrequenz bei der Abarbeitung von FOLGE möglichst genau einzustellen, durchläuft das Programm nach der Umspeicherung eines jeden Bildes zwei verschachtelte Warteschleifen. Die Anzahl der Schleifendurchläufe wird von W1 (äußere Schleife) und W2 (innere Schleife) angegeben. Die Gesamtzeit der Warteschleife ergibt sich somit aus W1×W2.
Im laufenden Programm kann der Wert von W1 und W2 durch Betätigung der Cursortasten (von 1 bis 255) geändert werden:

Cursor links: (langsamer)	W1 := W1 + 1
Cursor rechts: (schneller)	W1 := W1 − 1
Cursor runter: (langsamer)	W2 := W2 + 1
Cursor hoch: (schneller)	W2 := W2 − 1

3. MEM
Wenn man mit SCME ein Bild im Speicher ablegen will, wird dessen Adresse durch MEM angegeben. Da nach Abarbeitung von SCME MEM auf das erste Byte nach dem Bild im Speicher zeigt,

ist die Angabe der Startadresse nur am Anfang eines Blockes notwendig. Es wird dazu #4000 verwendet.

4. TAB

Zur Geschwindigkeitssteigerung des Programms wird eine Adresstabelle benutzt, die die Adressen der 1. Bytes jeder verwendeten Pixelzeile im IRM enthält. Sie wird beim Aufruf von INIT ab der in TAB angegebenen Adresse angelegt. Die Tabelle hat eine Länge von $2\times(YE-YA+1)\times 8$ Bytes, das heißt, für jede Pixelzeile 2 Byte. Man kann dafür am besten das Ende des BASIC-Speichers mit CLEAR herabsetzen und dann TAB auf den Beginn des freien Speicherbereiches zeigen lassen.

5. DATE

Das Vorführen der Bildsequenzen kann sehr variabel gestaltet sein. Dazu zeigt DATE auf den Anfang einer weiteren Tabelle, die eine Menge von Steuerbytes enthält.

Aufbau der Tabelle ab (DATE):

Byte	Bedeutung	
1.	: Anzahl der 16-K-Blöcke (n)	
2.	: Modulsteckplatz	(1. Block)
3.	: Steuerbyte	(1. Block)
4.	: Anzahl der Bilder	(1. Block)
5.	: Modulsteckplatz	(2. Block)
6.	: Steuerbyte	(2. Block)
7.	: Anzahl der Bilder	(2. Block)

usw.

3* n—1:	Modulsteckplatz	(n-ter Block)
3* n	: Steuerbyte	(n-ter Block)
3* n+1:	Anzahl der Bilder	(n-ter Block)

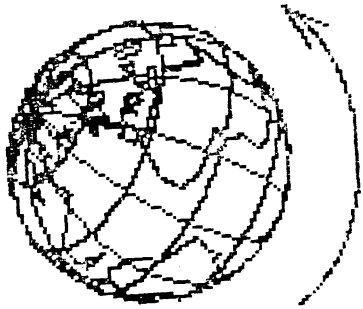
3* n+2: Endemarke:
0 – einmaliges Anzeigen aller Bilder
1 – Neustart von vorn

4. Assembler-Listing

Ich gebe hier das vollständige GENS-Listing (s. S. 27) an.
Die Kommentare müssen natürlich nicht mit übernommen werden; ich hoffe aber, daß sie das Listing verständlicher machen.

5. Beispiel »Drehende Erde«

Aus der Fülle der Anwendungsmöglichkeiten möchte ich als Beispiel ein einfach zu realisierendes herausgreifen. Das Ziel ist die Darstellung der rotierenden Erde.



Dabei gehe ich von dem Vorhandensein eines 64-K-Moduls aus. Um den Eindruck einer sich gleichförmig um ihre Achse drehenden Erde hervorzurufen, genügt es für eine volle Drehung um 360°, 80 Ansichten (Phasen) der Erde nacheinander vorzuführen, wobei aufeinanderfolgende Ansichten sich um 4,5 Längengrade unterscheiden. Bei 64K kommen also in jeden 16-K-Block 20 Bilder, die jedes 800 Byte einnehmen können. Die Erde kann damit 10*10 Zeichen groß werden. Allerdings müßten wir nun erst einmal 80 Bilder der Erde zeichnen. Glücklicherweise ist diese Arbeit bereits getan. Zur Erzeugung der einzelnen Phasen ziehe ich ein Programm heran, das allgemein bekannt ist. Es handelt sich um das Programm GLOBUS aus »BASIC-Einmaleins des Programmierens« von U. Grote und H. Völz, URANIA-Extraausgabe 1986, das auch durch einen Rundfunk-BASIC-Kurs weit verbreitet wurde.

```
                        10 ;MASCHINENPROGRAMM ZUR SPEICHERUNG
                        20 ;UND AUSGABE EINER BILDFOLGE
                        30
                        40 ;(C)1988 U.Girlich, Leipzig
                        50
                        60
0000                    70         ORG  0          ;Startadresse: #0000
                        80
                        90 ;Initialisierung
                       100
0000 CDE400            110 INIT   CALL BEGIN      ;Anfang
0003 3AF500            120        LD   A,(XA)     ;XA in E
0006 5F                130        LD   E,A
0007 3AF600            140        LD   A,(XE)     ;DX aus XE-XA+1 bilden
000A 93                150        SUB  E
000B 3C                160        INC  A
000C 32ED00            170        LD   (DX),A
000F 3AF300            180        LD   A,(YA)     ;YA in Pixelkoordinaten
0012 17                190        RLA             ;umwandeln (YA*8)
0013 17                200        RLA
0014 17                210        RLA
0015 57                220        LD   D,A        ;und in D
0016 3AF400            230        LD   A,(YE)     ;DY aus YE*8-YA*8+8 bilden
0019 17                240        RLA
001A 17                250        RLA
001B 17                260        RLA
001C 92                270        SUB  D
001D C608              280        ADD  A,8
001F 32EE00            290        LD   (DY),A
0022 47                300        LD   B,A        ;DY Zeilen
0023 2AF700            310        LD   HL,(TAB)   ;HL: Tabellenzeiger
0026 D5                320 M01    PUSH DE         ;Koordinaten retten
0027 E5                330        PUSH HL         ;Tabellenzeiger retten
0028 EB                340        EX   DE,HL      ;HL:=DE
0029 CD03F0            350        CALL #F003      ;Pixelkoordinaten berechnen
002C 34                360        DEFB #34
002D EB                370        EX   DE,HL      ;DE:=HL
002E E1                380        POP  HL         ;Tabellenzeiger holen
002F 73                390        LD   (HL),E     ;Pixeladresse in Tabelle
0030 23                400        INC  HL         ;eintragen und
0031 72                410        LD   (HL),D     ;Tabellenzeiger weiterstellen
0032 23                420        INC  HL
0033 D1                430        POP  DE         ;Koordinaten holen
0034 14                440        INC  D          ;Y weiterstellen
0035 10EF              450        DJNZ M01        ;nächste Zeile
0037 F1                460 END    POP  AF         ;IRM-Zustand holen
0038 D388              470        OUT  (#88),A    ;IRM schalten
003A C9                480        RET             ;zurück ins rufende Programm
                       490
                       500 ;Bildfolge
                       510
003B CDE400            520 FOLGE  CALL BEGIN      ;Anfang
003E 2AF900            530 M02    LD   HL,(DATE);HL: Datenzeiger
0041 4E                540        LD   C,(HL)     ;Blockanzahl in C
0042 23                550        INC  HL         ;Datenzeiger weiterstellen
0043 C5                560 M03    PUSH BC         ;Zähler retten
0044 46                570        LD   B,(HL)     ;Modulsteckplatz in B
0045 23                580        INC  HL         ;Datenzeiger weiterstellen
0046 0E80              590        LD   C,#80      ;I/O-Adresse der Modulsteuerung
0048 7E                600        LD   A,(HL)     ;Steuerbyte in A
0049 23                610        INC  HL         ;Datenzeiger weiterstellen
004A ED79              620        OUT  (C),A      ;Modul anschalten
004C C1                630        POP  BC         ;Zähler holen
004D 46                640        LD   B,(HL)     ;Bilder pro Block in B
004E 23                650        INC  HL         ;Datenzeiger weiterstellen
004F 110040            660        LD   DE,#4000   ;Startadresse des 1. Bildes.
0052 ED53EF00          670        LD   (MEM),DE   ;nach MEM
0056 E5                680 M04    PUSH HL         ;Datenzeiger retten
0057 C5                690        PUSH BC         ;Zähler retten
0058 CD9000            700        CALL MESC       ;Bild in IRM
005B C1                710        POP  BC         ;Zähler holen
005C E1                720        POP  HL         ;Datenzeiger holen
005D 38D8              730        JR   C,END      ;BRK? ja: Ende
005F 10F5              740        DJNZ M04        ;nächstes Bild
0061 0D                750        DEC  C          ;Blockanzahl verringern
0062 20DF              760        JR   NZ,M03     ;nächster Block
0064 7E                770        LD   A,(HL)     ;Endemarke in A

0065 B7                780        OR   A          ;0?
0066 20D6              790        JR   NZ,M02     ;nein: Sprung zum Anfang
0068 18CD              800        JR   END        ;Ende
                       810
                       820 ;Screen -> Memory
                       830
006A CDE400            840 SCME   CALL BEGIN      ;Anfang
006D 2AF700            850        LD   HL,(TAB)   ;HL: Tabellenzeiger
0070 3AEE00            860        LD   A,(DY)     ;DY Zeilen
0073 47                870        LD   B,A
0074 3AED00            880        LD   A,(DX)     ;DX Byte pro Zeile
0077 4F                890        LD   C,A
0078 5E                900 M05    LD   E,(HL)     ;Pixeladresse aus Tabelle
0079 23                910        INC  HL         ;in DE und
007A 56                920        LD   D,(HL)     ;Tabellenzeiger weiterstellen
007B 23                930        INC  HL
007C E5                940        PUSH HL         ;Tabellenzeiger retten
007D 2AEF00            950        LD   HL,(MEM)   ;HL: Speicherzeiger
0080 EB                960        EX   DE,HL      ;Blockverschiebung
```

```
0081 C5          970         PUSH BC         ;C Bytes von DE nach HL
0082 0600        980         LD   B,0
0084 EDB0        990         LDIR
0086 C1         1000         POP  BC
0087 EB         1010         EX   DE,HL
0088 22EF00     1020         LD   (MEM),HL  ;Speicherzeiger weg
008B E1         1030         POP  HL        ;Tabellenzeiger holen
008C 10EA       1040         DJNZ M05       ;nächste Zeile
008E 18A7       1050         JR   END       ;Ende
                1060
                1070 ;Memory -> Screen
                1080
0090 2AF700     1090 MESC    LD   HL,(TAB)  ;HL: Tabellenzeiger
0093 3AEE00     1100         LD   A,(DY)    ;DY Zeilen
0096 47         1110         LD   B,A
0097 3AED00     1120         LD   A,(DX)    ;DX Bytes pro Zeile
009A 4F         1130         LD   C,A
009B 5E         1140 M06     LD   E,(HL)    ;Pixeladresse aus Tabelle
009C 23         1150         INC  HL        ;in DE und
009D 56         1160         LD   D,(HL)    ;Tabellenzeiger weiterstellen
009E 23         1170         INC  HL
009F E5         1180         PUSH HL        ;Tabellenzeiger retten
00A0 2AEF00     1190         LD   HL,(MEM)  ;HL: Speicherzeiger
00A3 C5         1200         PUSH BC        ;Blockverschiebung
00A4 0600       1210         LD   B,0       ;C Bytes von HL nach DE
00A6 EDB0       1220         LDIR
00A8 C1         1230         POP  BC
00A9 22EF00     1240         LD   (MEM),HL  ;Speicherzeiger weg
00AC E1         1250         POP  HL        ;Tabellenzeiger holen
00AD 10EC       1260         DJNZ M06       ;nächste Zeile
                1270
                1280 ;Pause
                1290
00AF 21F100     1300         LD   HL,W1     ;HL: Zeiger auf W1
00B2 46         1310         LD   B,(HL)    ;B: Zähler äußere Schleife
00B3 23         1320 M07     INC  HL        ;Zeiger auf W2
00B4 4E         1330         LD   C,(HL)    ;C: Zähler innere Schleife
00B5 2B         1340         DEC  HL        ;Zeiger auf W1
00B6 DDCB0846   1350 M08     BIT  0,(IX+8)  ;Taste gedrückt?
00BA 2816       1360         JR   Z,M09     ;nein: weiter
00BC 3AFD01     1370         LD   A,(#01FD) ;Taste lesen
00BF DDCB0886   1380         RES  0,(IX+8)  ;quittieren
00C3 FE03       1390         CP   3         ;BRK?
00C5 37         1400         SCF            ;Carry-Flag setzen
00C6 C8         1410         RET  Z         ;ja: zurück
00C7 D608       1420         SUB  8         ;Tastencode um 8 verringern
00C9 CDD900     1430         CALL M10       ;Code 8,9; W1 setzen
00CC 23         1440         INC  HL        ;Zeiger auf W2
00CD 3D         1450         DEC  A         ;Tastencode verringern
00CE CDD900     1460         CALL M10       ;Code 10,11; W2 setzen
00D1 2B         1470         DEC  HL        ;Zeiger auf W1
00D2 0D         1480 M09     DEC  C         ;innere Schleife weiter
00D3 20E1       1490         JR   NZ,M08
00D5 10DC       1500         DJNZ M07       ;äußere Schleife weiter
00D7 B7         1510         OR   A         ;Carry-Flag rücksetzen
00D8 C9         1520         RET            ;zurück
                1530
00D9 2005       1540 M10     JR   NZ,M13    ;W1:
00DB 34         1550 M11     INC  (HL)      ;8: erhöhen
00DC C0         1560         RET  NZ        ;9: verringern
00DD 35         1570 M12     DEC  (HL)      ;bzw. W2:
00DE 28FB       1580         JR   Z,M11     ;10: erhöhen
00E0 3D         1590 M13     DEC  A         ;11: verringern
00E1 28FA       1600         JR   Z,M12
00E3 C9         1610         RET            ;zurück
                1620
00E4 E1         1630 BEGIN   POP  HL        ;Rücksprungadresse in HL

00E5 DB88       1640         IN   A,(#88)   ;IRM-Zustand lesen
00E7 F5         1650         PUSH AF        ;IRM-Zustand retten
00E8 CBD7       1660         SET  2,A       ;IRM an...
00EA D388       1670         OUT  (#88),A   ;schalten
00EC E9         1680         JP   (HL)      ;zurück
                1690
00ED 00         1700 DX      DEFB 0         ;Breite des Fensters (Bytes)
00EE 00         1710 DY      DEFB 0         ;Höhe des Fensters (Pixel)
00EF 0000       1720 MEM     DEFW 0         ;Zeiger auf Bildanfang
00F1 00         1730 W1      DEFB 0         ;Länge äußere Warteschleife
00F2 00         1740 W2      DEFB 0         ;Länge innere Warteschleife
00F3 00         1750 YA      DEFB 0         ;1. Zeile des Fensters
00F4 00         1760 YE      DEFB 0         ;letzte Zeile des Fensters
00F5 00         1770 XA      DEFB 0         ;1. Spalte des Fensters
00F6 00         1780 XE      DEFB 0         ;letzte Spalte des Fensters
00F7 0000       1790 TAB     DEFW 0         ;Beginn der Adresstabelle
00F9 0000       1800 DATE    DEFW 0         ;Beginn der Datentabelle

Pass 2 errors: 00
```

Natürlich muß einiges in diesem Programm geändert werden:

Löschen der Zeilen 20 bis 60
80 bis 120
135 bis 290
450 bis 485.

Zusätzlich muß folgendes eingegeben werden:

```
10 CLEAR0,16223:COLOR7,0:WINDOW0,31,0,39:CLS
12 PRINT"Band starten! MC laden.":BLOAD
15 GOTO140
70 LINEX1,Y1,X2,Y2,7
140 POKE243,11:POKE244,20:POKE245,15:POKE246,24
150 DOKE247,16224:CALL0
160 BM=PI/180:A=-30*BM:B=-23*BM:C=30*BM:X0=159:Y0=127:H=.5:R=39
170 S1=SIN(A):S2=SIN(B):C1=COS(A):C2=COS(B)
180 CW=360
190 CLS:PRINTAT(2,2);"CW=";CW;" "
200 CIRCLEX0,Y0,R,7
210 C=-(CW+90)*BM:S3=SIN(C):C3=COS(C)
220 RESTORE
445 IFCW<=180THEN460
450 IFCW>=274.5THENST=67:ELSEST=3
455 GOTO465
460 IFCW>=94.5THENST=195:ELSEST=131
465 SWITCH8,ST
470 M=80-CW/4.5:M=M-20*INT(M/20):M=M*800+16384:DOKE239,M
475 CALL106:CW=CW-4.5
480 IFCW>0THEN190
485 END
```

Außerdem erweist es sich als günstig, die Schrittweite in den Zeilen 400 und 430 von 2,5° auf 5° zu erhöhen.
Die Programmierung erscheint zunächst etwas umständlich. Vor allem das ständige SWITCHen des Moduls und das laufende Beschreiben von MEM ist nicht unmittelbar einzusehen.

Auf diese Weise kann aber an jeder beliebigen Stelle die rund 16 Stunden dauernde Berechnung unterbrochen werden, um z.B. schon fertige Blöcke auf die Kassette zu sichern. Beim Neustart ist dann nur der Winkel, bei dem die Berechnung fortgesetzt werden soll, in Zeile 180 einzutragen und das Programm mit RUN zu starten, ohne sich um Startadressen oder Modulschaltungen kümmern zu müssen.
Nachdem alle 4 Blöcke auf Kassette vorliegen, ist darauf zum Ansehen der sich drehenden Erde folgendes Programm einzugeben:

```
10 CLEAR0,16223:COLOR7,0:WINDOW0,31,0,39:CLS
20 PRINT"Band starten!"
30 PRINT"MC":BLOAD
40 PRINT"BLOCK 1":SWITCH8,67:BLOAD
50 PRINT"BLOCK 2":SWITCH8,3:BLOAD
60 PRINT"BLOCK 3":SWITCH8,195:BLOAD
70 PRINT"BLOCK 4":SWITCH8,131:BLOAD
80 CLS
90 RESTORE
100 FORI=241TO264
110 READA:POKEI,A
120 NEXTI
130 CALL0
140 CALL59
150 END
160 DATA126,24,11,20,15,24,96,63,251,0
170 DATA4,8,67,20,8,3,20,8,195,20,8,131,20,1
```

Es liest dann selbständig alle Bilder ein und setzt die Variablen. Die Rotationsgeschwindigkeit ist auf 10 Hz eingestellt, die Erde dreht sich also in 8s einmal um sich selbst. Die maximalmögliche Geschwindigkeit beträgt bei

dieser Fenstergröße über 60 Hz, was aber wegen der Bildwiederholfrequenz des Fernsehers keinen akzeptablen Eindruck vermittelt.

6. Veränderungen und Erweiterungsmöglichkeiten

6.1. Vermeiden des Bildschirmflackerns

Es ist sicher eine interessante Ergänzung, das Flackern auf dem Bildschirm zu unterdrücken. Es flackert bei IRM-Zugriffen an genau der Stelle, wo sich der Elektronenstrahl der Fernsehröhre gerade befindet. Flackern bedeutet immer das Setzen der Vordergrundfarbe auf Hintergrundfarbe. Bei dem vorgestellten Beispiel bedeutet dies: zur Unterdrückung des Flackerns genügt es, die Umlagerung des Bildes zeitlich so mit dem Fernsehsignal zu synchronisieren, daß sich der Elektronenstrahl oberhalb bzw. unterhalb der Erdkugel befindet. Das ist sehr einfach, denn der Blink-CTC (Port #8E) wird genau in dem Moment weitergestellt, wenn der Elektronenstrahl ein neues Bild beginnt. Man muß also auf eine Veränderung des Zustandes des CTC warten (IN A,(#8E)). Es folgt eine Warteschleife, die genau dann beendet ist, wenn der Elektronenstrahl die untere Kante der Erde erreicht hat. Es muß dann die Umlagerung stattfinden, die aber in der jetzigen Form zu lange dauert, so daß der obere Rand der Erde wieder flackert. MESC muß also noch zeitoptimiert werden. Dann könnte jedes zweite Fernsehbild verwendet werden. Das ergibt 25 Hz. W1 kann nun genutzt werden, um anzugeben, wie viele Fernsehbilder bis zur nächsten Phase der Drehung abgewartet werden sollen.

6.2. Einsatz mehrerer Moduln

In der vorliegenden Version des Programms ist es nur möglich, mit einem Speichererweiterungsmodul (16, 64 bzw. 256KByte) zu arbeiten, da der Modul nach seiner Verwendung angeschaltet bleibt und so wegen der Prioritätskette alle anderen Moduln sperrt, die eine höhere Modulsteckplatznummer haben. Um mehrere Moduln für die Speicherung von Bildern zu verwenden, muß eine Ergänzung im Assemblerlisting vorgenommen werden:
Mit dieser Ergänzung schaltet das Programm nach jedem Block den eben verwendeten Modul aus; es sind also auch alle in der Prioritätskette folgenden Moduln nutzbar. Dadurch wird das System beträchtlich erweiterbar.

6.3. Fenstertechnik

Eine völlig andere Nutzung der Routinen SCME und MESC besteht in der Entwicklung einer »echten« Fenstertechnik, bei der dann nach Schließung eines Fensters der alte Hintergrund

```
650   DEC   HL       ;Datenzeiger auf Steuerbyte
655   DEC   HL       ;Datenzeiger auf Modulsteckplatz
741   PUSH  BC       ;Zähler retten
742   LD    B, (HL)  ;Modulsteckplatz in B
743   INC   HL       ;Datenzeiger auf Steuerbyte
744   INC   HL       ;Datenzeiger auf Bildanzahl
745   INC   HL       ;Datenzeiger weiterstellen
746   LD    C, #80   ;I/O-Adresse der Modulsteuerung
747   XOR   A        ;A:=0
748   OUT   (C), A   ;Modul ausschalten
749   POP   BC       ;Zähler holen
```

bzw. andere Fenster wieder zum Vorschein gelangen. Damit nicht zuviel Speicherplatz verwendet wird, sollte man Grafik und Text getrennt abspeichern und die Grafik dabei gleich komprimieren. Die einfachste Methode dazu besteht darin, daß statt mehrerer Nullen nur eine Null als Marke und danach die Anzahl der Nullen abgespeichert wird. Im allgemeinen bringt das schon eine erhebliche Speicherplatzersparnis.
Dazu muß man aber das Programm noch dahingehend erweitern, daß das verwendete Window auch über die Grenze zwischen der 31. und 32. Spalte des Bildschirmes gehen kann.
Ich glaube, dieses Programm zeigt ganz deutlich, daß die meist wegen ihrer geringen Geschwindigkeit belächelten Kleincomputer doch recht gut in der Lage sind, auch kompliziert erscheinende Probleme zu bewältigen.

Autor:
Uwe Girlich
Leipzig

Korrekturhinweis zu Heft 10

Beitrag
FORTH – ein Softwarekonzept für Mikro- und Minicomputer

Bedingt durch den Schriftsatz in der Druckerei sind im Beitrag einige Inkorrektheiten entstanden, die gerade bei FORTH kritisch sind.

1. Alle in Tabelle 1 aufgeführten FORTH-Worte müssen zusammengeschrieben werden (ohne Leerzeichen).
2. In allen Beispielen müssen die FORTH-Worte durch mindestens ein Leerzeichen getrennt sein.
3. Auf Seite 17 im zweiten Beispiel muß es korrekt heißen:
   ."DIVISOR=Ø" ; ." ist der Operator zur Ausgabe einer Zeichenkette, die mit " endet.

Wir bitten, diese Inkorrektheiten zu entschuldigen.

Elektronische Schreibmaschine S 6005 als Drucker

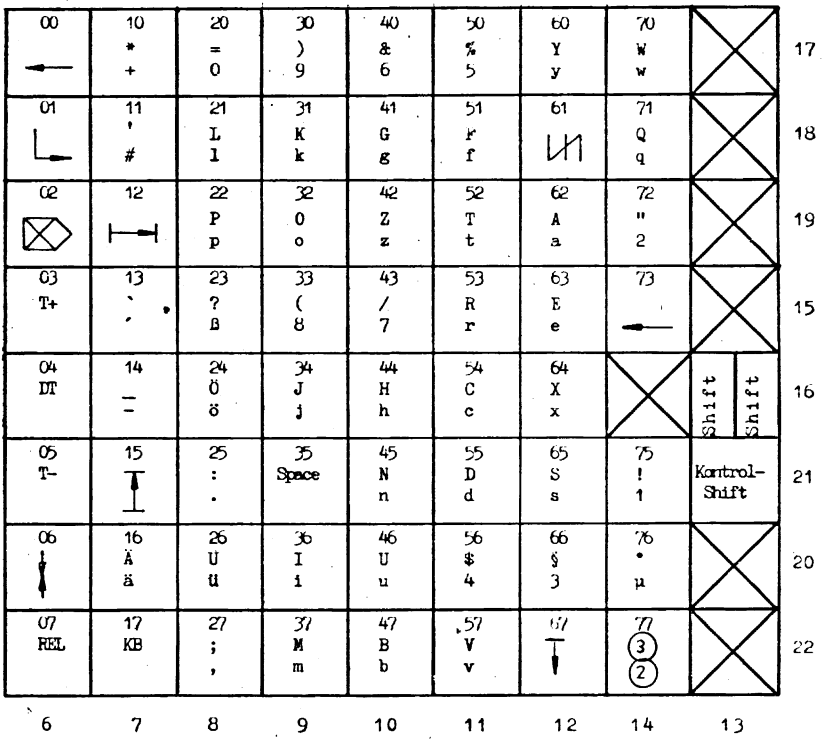
Früher oder später entsteht bei jedem Nutzer eines Heimcomputers der Wunsch, möglichst zu drucken. Sowohl das Drucken des Listings von Programmen als auch die Nutzung eines Textverarbeitungssystems verlangen nach dieser Möglichkeit. Da noch keine speziellen Drucker im Handel angeboten werden, kann sich der Amateur mit dem Umbau eines Fernschreibers oder einer elektronischen Schreibmaschine behelfen. Inzwischen werden zwar auch elektronische Schreibmaschinen mit einer genormten Schnittstelle zur Datenübertragung hergestellt, für den Amateur dürften diese allerdings nur schwer beschaffbar sein. Außerdem sind diese Geräte auch wesentlich teurer als eine einfache elektronische Schreibmaschine. Diesem Umstand folgend, soll der Umbau der elektronischen Schreibmaschine S 6005 beschrieben werden.

1. Funktion der Schreibmaschine

Die Schreibmaschine besteht im wesentlichen aus der Tastatur, dem Druckwerk und einer Papiertransporteinrichtung. Für den Umbau zum Drucker ist nur die Tastatur als Eingabeschnittstelle von Interesse.
Die Tastaturen sind matrixförmig organisiert. Der Rechner der Schreibmaschine legt beispielsweise an die Spalten der Matrix nacheinander ein Signal und kontrolliert, ob dieses Potential an den Zeilen wieder auftritt. Ist dies der Fall, so läßt sich aus der aktivierten Spalte und Zeile, an der das Signal auftritt, eindeutig die gedrückte Taste bestimmen. Nun muß noch dafür gesorgt werden, daß jede Tastenbetätigung auch sicher erkannt wird, denn bekanntlich prellen mechanische Tasten sehr stark. Das kann erfolgen, indem der Rechner sich bei der ersten Abfrage nur die gedrückte Taste merkt. Nach einer kleinen Pause, in der die Prellvorgänge abgeklungen sind, kontrolliert er die Tastatur erneut. Erkennt er dabei, daß noch immer die gleiche Taste gedrückt ist, stellt er deren Zeichen zur Verarbeitung bereit, andernfalls löscht er den Merkspeicher und beginnt mit einem neuen Abfragezyklus. Wichtig ist ferner, daß die Schreibmaschine vor Auswertung der Zeichentasten eventuell gedrückte Steuertasten erkennt.
Diese läßt sich dadurch erreichen, daß jede Abfrage mit der Kontrolle dieser Steuertasten beginnt. Bild 1 zeigt den Aufbau der Tastaturmatrix der S 6005.

2. Mechanischer Umbau der Schreibmaschine

Der mechanische Umbau beschränkt sich auf die Anbringung eines Durchbruchs für den Steckverbinder und die Montage eines direkten 58poligen Steck-



Anschlußbezeichnung von oben links

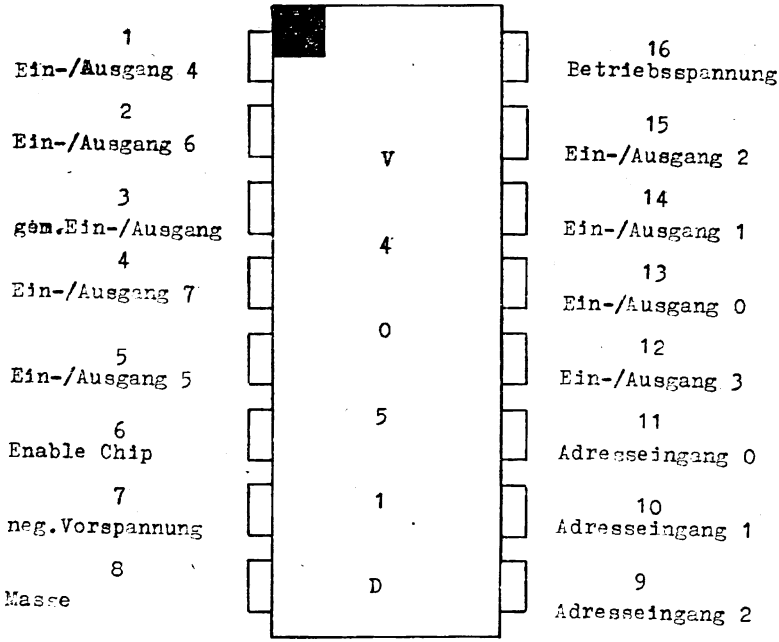
Feldinhalt Kodeadresse
Symbol mit Shift
Symbol ohne Shift

Bild 1. Aufbau der Tastaturmatrix für die elektronische Schreibmaschine S 6005

verbinders am Prüfanschluß für die Tastatur. Es ist dabei darauf zu achten, daß die Anschlüsse des Steckverbinders soweit gekürzt werden, daß sie die Bewegung des Druckwagens nicht behindern. Vor der Montage sollte der Steckverbinder verdrahtet und das Flachbandkabel so geführt werden, daß der Druckwagen sich ungehindert bewegen kann. Blickt man von vorn auf die Schreibmaschine, so befindet sich der Anschluß 1 rechts oben in der Ecke.

3. Steuerelektronik

Die Steuerelektronik muß elektronisch den Druck einer Taste simulieren, das heißt eine leitende Verbindung zwischen Zeilen und Spalten herstellen. Als Herzstück der Schaltung werden Analogmultiplexer in CMOS-Technik U 4051 verwendet. Bild 2 zeigt Anschlußbelegung und Funktionsschaltbild dieser integrierten Schaltkreise.
Bild 3 zeigt die Gesamtschaltung der Ansteuerelektronik. Der EPROM dient zur Umcodierung der vom Rechner gesendeten ASCII-Zeichen in den »Schreibmaschinencode«. Es wurde der hardwaremäßigen Umcodierung der Vorzug gegeben, da der Speicherplatz in Heimcomputern oft sehr knapp ist und nicht immer die Implementierung einer län-



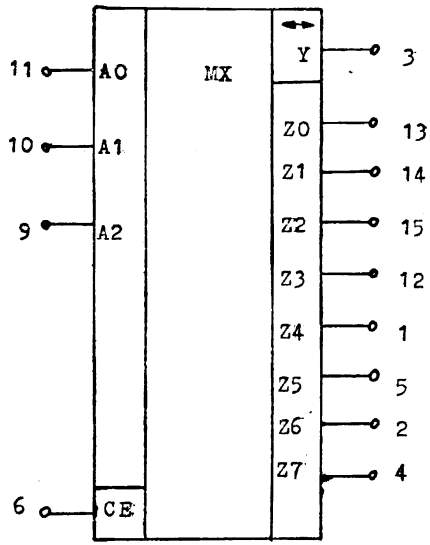


Bild 2. Anschlußbelegung und Schaltsymbol U 4051

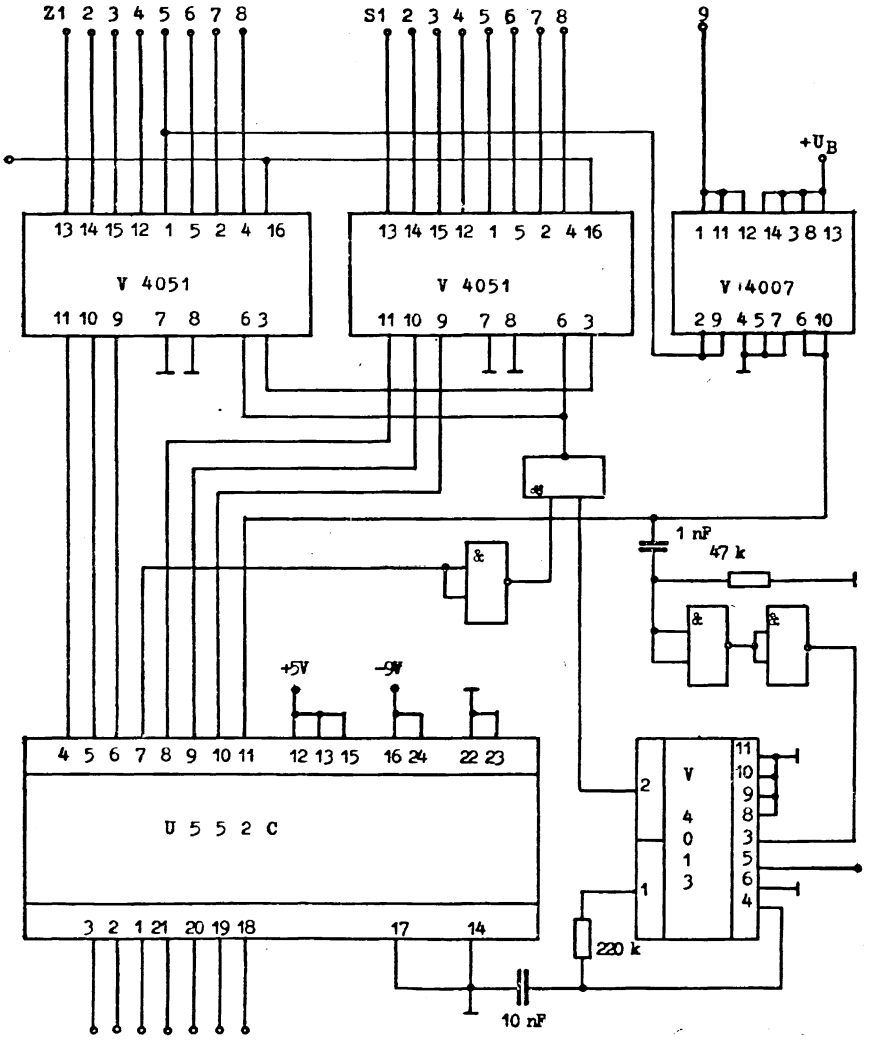


Bild 3. Schaltung Ansteuerelektronik

geren Druckroutine erlaubt. Als Eingangssignal liegt an den Adreßeingängen der ASCII-Code an. An den Ausgängen 1 bis 3 liegt die Zeilenadresse und an den Ausgängen 5 bis 7 die Spaltenadresse des zu druckenden Zeichens auf der Tastaturmatrix. Der V 4007 übernimmt die Shift-Umschaltung. Durch ein High-Signal am Ausgang 8 des EPROMs wird dieser durchgeschaltet. Gleichzeitig wird über ein Differenzierglied die Vergrößerungsstufe (V 4013) aktiviert. Erst nach einer Verzögerung von etwa 1 ms werden die Schalter für die Matrix freigegeben. Damit ist der Vorrang der Shifttaste gegenüber den Zeichentasten gewährleistet. Der Ausgang 4 wird als Statusausgang verwendet. Ist er nicht akti-

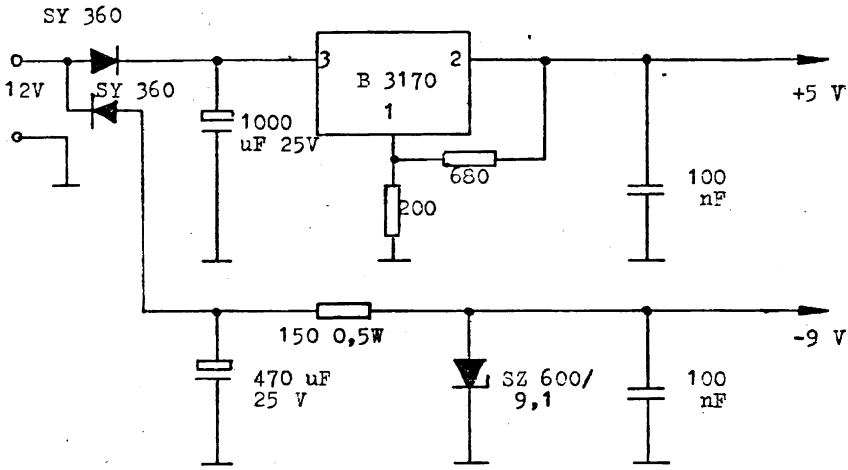


Bild 4. Stromversorgung der Koppelbaugruppe

viert, so wird die Matrix nicht geschaltet. Die gesamte Schaltung findet zusammen mit der Stromversorgung nach Bild 4 auf einer Leiterplatte im Format 95 mm × 170 mm Platz. Ein 15poliger Steckverbinder dient als Verbindung zur PIO. Zur Schreibmaschine hin ist ein 26poliger Steckverbinder als Schnittstelle vorgesehen. Obwohl die Verbindungsleitung zur Schreibmaschine reichlich 1,5 m lang ist, traten keinerlei Probleme auf.

4. Software

Das hier vorgestellte Programm (s. S. 37) realisiert die Ausgabe eines Zeichens auf dem Drucker. Als Steuerzeichen wird nur das Carriage Return/Line Feed (ODH) ausgewertet. Das Programm läuft auf dem Z 1013 bei einer Taktfrequenz von 2 MHz. Bei anderen Rechnern sind die Portadresse und die Zeitkonstante zu ändern. Die Zeitkonstanten sind experimentell zu ermitteln. Wird ein Zeichen mehrfach hintereinander gedruckt, so ist die Ausgabezeit zu lang. Der Drucker führt die Repeatfunktion aus. Ist die Zeit zu klein, so werden einzelne Zeichen »verschluckt«. Bei der Bemessung der Rücklaufzeit ist zu berücksichtigen, daß die S 6005 nur über einen Puffer von 5 Zeichen verfügt. Auch beim Rücklauf vom äußersten rechten Rand darf dieser Puffer nicht überfüllt werden. Dadurch entsteht bei kurzen Zeilen eine kleine Pause am Zeilenanfang. Durch ein Druckprogramm, welches die Anzahl der Zeichen je Zeile zählt, ließe sich das Drucken noch beschleunigen. Inwieweit dieser Aufwand lohnt, mag jeder Nutzer selbst entscheiden.

Mit diesem Druckprogramm kann maximal eine Druckgeschwindigkeit von 8 bis 10 Zeichen/s erreicht werden. Eine höhere Geschwindigkeit kann zur elektrischen Überlastung der Schreibmaschine führen.

Die vorgestellte Schaltung und die Druckroutine werden vom Autor als Bestandteil eines Z 1013-Rechnersystems seit 1986 genutzt. Die Druckroutine ist als Unterprogramm im BASIC-Interpreter und Text-Editor eingearbeitet. Bei diesen Programmen wurde

```
8000 7F 7F 7F 7F 7F 7F 7F 7F 00 73 67 15 02 01 7F 7F  ........sg.....  05E8
8010 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F  ................  07F0
8020 35 F5 F2 11 D6 D0 C0 91 B3 B0 90 10 27 14 25 C3  5...........'.%.  084A
8030 20 75 72 66 56 50 40 43 33 30 A5 A7 35 A0 35 A3   urfVP@C30..5.5.  05F2
8040 76 E2 C7 D4 D5 E3 D1 C1 C4 B6 B4 B1 A1 B7 C5 B2  v...............  0BEB
8050 A2 F1 D3 E5 D2 C6 D7 F0 E4 E0 C2 35 35 35 F6 94  ...........555..  0B59
8060 7F 62 47 54 55 63 51 41 44 36 34 31 21 37 45 32  .bGTUcQAD641!7E2  0474
8070 22 71 53 65 52 46 57 70 64 60 42 35 35 35 35 06  "qSeRFWpdB5555.   048A
8080 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F  ................  07F0
8090 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F  ................  07F0
80A0 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F  ................  07F0
80B0 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F  ................  07F0
80C0 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F  ................  07F0
80D0 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F  ................  07F0
80E0 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F  ................  07F0
80F0 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F  ................  07F0
```

Tafel 1. Programmiervorschlag EPROM

ein Einsprung geschaffen, der vor dem Start des Programms die Druckerschnittstelle initialisiert. Das ist nötig, um ein Blockieren der Tastatur der Schreibmaschine zu vermeiden. Sollte die Shiftumschaltung nicht zuverlässig funktionieren, so kann die Ursache im zu hohen Kanalwiderstand des V 4007 liegen. Ein zweiter parallelgeschalteter V 4007 schafft hier Abhilfe. Tafel 1 zeigt noch einen Vorschlag zur Programmierung des EPROMs.

Autor:
Andreas Köhler
Fertigungstechnologe im
RAW „Otto Grotewohl"
Dessau

Programmlisting

```
Initialisierung der PIO
1000        3E  CF      LD A, 0CFH
1002        D3  00      OUT PORT 00          PIO im Bit E/A-Mode
1004        3E  00      LD A, 000H
1006        D3  00      OUT PORT 00          8 Ausgänge
1008        D3  01      OUT PORT 01          Keine Taste »betätigt«
100A        C9          RET
Druck eines Zeichens aus dem A-Register
0DH         einziges Steuerzeichen
100B        D3  01      OUT PORT 01          »Druck der Zeichentaste«
100D        CD  1C 10   CALL ZEIT KURZ
1010        FE  0D      CMP 0DH              Steuerzeichen Wagenrücklauf?
1012        20  03      JRNZ M1              nein!
1014        CD  27 10   CALL ZEIT LANG
1017 M1:    AF          XOR A                »Taste loslassen«
1018        CD  1C 10   CALL ZEIT KURZ
101B        C9          RET                  Zum Hauptprogramm nächstes Zeichen
                                             holen
```

Fortsetzung auf Seite 64

Grafikbaugruppe für den Mikrorechnerbausatz Z 1013

Einleitung

Die hier vorgestellte Baugruppe ermöglicht eine grafische Auflösung von 256×256 Bildpunkten. Der normale Textmodus kann weiterhin verwendet werden.

1. Aufbau

Als Bildpunktspeicher findet ein 8 KByte SRAM U 6264 Verwendung (Bild 1). Lösungen unter Verwendung der IC 6116 oder U 214 D sind auch denkbar, bedingen jedoch einen entsprechenden Mehraufwand. Der RAM liegt im selben Adreßbereich wie der Bildschirm-RAM im Grundgerät (EC00 – EFFFH); die Umschaltung zwischen dem Bildpunkt- und dem Text-RAM erfolgt über D 4. Dazu machte es sich nötig, die /CS- bzw. /OE-Anschlüsse der

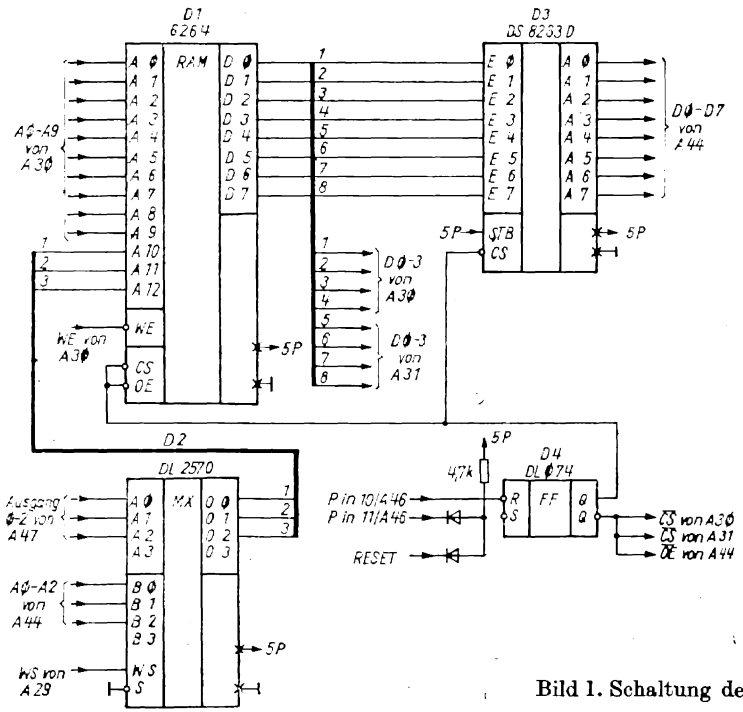


Bild 1. Schaltung der Grafikbaugruppe

IC A 30, A 31 und A 44 (Text-RAM und Zeichengenerator) von Masse zu trennen und entsprechend mit dem Flip-Flop zu verbinden. Nach RESET bzw. einem L-Impuls an Pin 11 des IC A 46 ist somit der Textmodus eingestellt, nach einem L-Impuls an Pin 10 des IC A 46 dagegen sind D 1 und D 3 selektiert, und dadurch werden die jeweiligen Bildinformationen direkt, also nicht über den Zeichengenerator, sondern über das Register D 3 ausgegeben. Die höchstwertigen 3 Bit des Adreßbusses des U 6264 werden unter Verwendung eines DL 257 D aus den niederwertigen 3 Bit des rechnerinternen Videozeilenzählers und aus 3 Anschlüssen des Registers A 47 gebildet.

2. Software

Bei der Betrachtung eines (8×8)-Pixel umfassenden Feldes des Bildschirmes läßt sich feststellen, daß die horizontale Aufteilung der Pixel der Nummer des entsprechenden Bits innerhalb eines Bytes entspricht und die vertikale Aufteilung durch die über A 47 ausgegebene 3-Bit-Information bestimmt wird. Die einzelnen Byte sind zunächst in einer Zeile hintereinander angeordnet. Aller 8 Pixel in der Vertikalen folgt dann jeweils die nächste Bytefolge des gerade über A 47 selektierten Blockes. Eine Initialisierung der Peripherie ist nicht nötig, da nur einfache Register und Dekoder Verwendung finden.
Bild 2 zeigt als Beispiel ein Programm zum Löschen des Grafikbildes unter Nutzung des Betriebssystemkommandos »KILL«. Da der Datentreiber invertierend ist, müssen dabei die 8 1-KByte-Blöcke mit FFH gefüllt werden.

Das Ansprechen der einzelnen RAM-Blöcke ist bei HC-BASIC nicht vollständig möglich, aber durch die Befehle OUT 8,8 bzw. OUT 8, 9 kann die Grafik aus- bzw. wieder eingeschaltet werden.

```
CLS:  LD A, 008H          ;Grafik ein
      OUT 008H
      XOR A
SUS:  OUT 008H            ;Block aktivieren
      PUSH AF
      LD HL, 0EC00H       ;Parameter laden
      LD (001BH), HL
      LD HL, 0EFFFH
      LD (001DH), HL
      LD HL, 000FFH
      LD (0023H), HL
      RST 020H            ;Funktion
                          "KILL"
      DB 011H
      POP AF
      INC A               ;nächsten Block
                          ansprechen
      CMP 008H
      JRNZ SUS
      LD A, 009H
      OUT 008H
      RET
```

Bild 2. Programm zum Löschen des Bildschirmes

Literatur

[1] Dokumentation MRB Z 1013. – Riesa: VEB Robotron Elektronik

[2] Streng, K.: Daten digitaler integrierter Schaltkreise. – Berlin: Militärverlag

[3] Hübner, U.: Grafikbildschirmsteuerung für Mikrorechner. – In: rfe. – Berlin 36 (1987) 10, – S. 634—638

Autoren:

Klaus Röbenack

Jork Hobohm

Eisleben

Programm zum chemischen Rechnen

In Heft 4 der „Kleinstrechner-Tips" stellte HEINS ein Programm zur Berechnung des Molekulargewichts chemischer Verbindungen vor. Darüber hinaus sind sicherlich Überlegungen über die Möglichkeiten sinnvoll, mit Hilfe eines KC 85/2 oder 85/3 Übungen für Schüler, die noch Schwierigkeiten beim chemischen Rechnen haben, zu gestalten.
Hier wird ein Programm (s. S. 43) vorgestellt, daß es erlaubt, Berechnungen für Massen und Volumina, wie sie in der Schulchemie der 7. und 8. Klassen auf der Grundlage chemischer Gleichungen aus der anorganischen Chemie vielfach angestellt werden, auszuführen. Der Schüler soll die Aufgabe mit der richtigen chemischen Gleichung (nicht in Ionenschreibweise!) vor sich haben, z.B.

$$Mg + 2\,HNO_3 \rightarrow Mg(NO_3)_2 + H_2.$$

Im Dialog fordert der Computer den Nutzer auf, eine Reihe von Eingaben zu machen.
Zunächst verlangt der Rechner, daß der Nutzer des Programms die Einheiten der gegebenen und gesuchten Größen analysiert und im Ergebnis dieser Analyse vier Aufgabentypen unterscheidet:

Massen gegeben und gesucht (1),
Volumen gegeben und gesucht (2),
Volumen gesucht (Masse gegeben) (3),
Volumen gegeben (Masse gesucht) (4)
(Zeilen 180 bis 240).

Gehören zu den gegebenen und gesuchten Größen nur Massen oder nur Volumina, gilt die Einheit der gegebenen Größe auch für das Ergebnis (Zeilen 1050 bzw. 1190). Gehört dagegen eine Masse *und* ein Volumen zu den gegebenen und gesuchten Größen, dann muß die gegebene Masse vor der Eingabe in Gramm und das gegebene Volumen in Liter umgerechnet worden sein. Die Einheit des Ergebnisses ist dann „Liter" (Zeile 1350) bzw. „Gramm" (Zeile 1510).
Der Computer fordert nun zur Eingabe der Formeln des gesuchten und des gegebenen Stoffes auf (Bild 1). Alle im wesentlichen zweiteiligen Verbindungen wie Oxide, Säure, Basen und Salze werden vom Rechner angenommen. Dazu sind die chemischen Zeichen der wichtigsten Elemente und auch die Säurereste im Rechner gespeichert (Zeilen 40 bis 110). Der Schüler hat die chemischen Zeichen und die Indizes für jedes Zeichen einzugeben. Die vor ihm liegende richtige chemische Gleichung bietet die richtigen Formeln an. Zu beachten ist nur, daß jeder Säurerest als Einheit betrachtet wird (wie bei den Übungen in der Schule üblich) und für einen Säurerest nur dann ein

```
CHEMISCHES RECHNEN
XXXXXXXXXXXXXXXXXXX

NUR MASSEN

  CHEMISCHE ZEICHEN UND INDIZES
  FUER DIE BERECHNUNG
  DER MOLAREN MASSEN:
GIB DAS CHEMISCHE ZEICHEN DES
STOFFES EIN, INDEM DU
DIE CHEMISCHEN ZEICHEN DER ELEMENTE
UND SAEUREREST DEN BUCHSTABEN A
UND B ZUORDNEST  UND DIE INDIZES
DEN BUCHSTABEN IA UND IB.

BEACHTE, DASS BEI DEN SAEUREREST
NUR DIE ANZAHL DER VOLLSTAENDIGEN
SAEUREREST EINGEGEBEN WIRD!!

GESUCHTER STOFF:      GEGEBENER STOFF:

  A = H                 A = MG
 IA =                  IA =
  B = NO3               B = NO3
 IB =                  IB = 2
```

Bild 1. Eingabe der chemischen Zeichen

Index eingegeben wird, wenn er mehrfach in der Formel enthalten ist.
Hat der Formelteil keinen Index, ist es gleichgültig, ob eine 1 eingegeben wird oder keine Eingabe erfolgt. Die Zeilen 500 und 510 verhindern, daß in Zeile 610 eine falsche molare Masse berechnet wird. Handelt es sich um ein chemisches Zeichen für ein Element, das in der chemischen Gleichung als Reaktionspartner auftritt, wird für den zweiten Teil der Formel keine Eingabe gemacht.
Weiterhin hat der Schüler die Faktoren, die in der vor ihm liegenden chemischen Gleichung vor den Formeln des gesuchten und des gegebenen Stoffes stehen, und die gegebene Masse bzw. das gegebene Volumen einzugeben. Kein Faktor bedeutet wieder die Eingabe einer 1 oder keine Eingabe.

Schließlich muß der Schüler seine Rechnung ausführen und auf Verlangen des Computers sein Ergebnis eingeben. Der Rechner bestätigt die Richtigkeit des Ergebnisses bei einem relativen Fehler unter 1 % oder verlangt ein neues Ergebnis (Zeilen 1600 bis 1690).
Die eingegebenen Werte und Zeichenketten werden wie folgt auf Variablen abgelegt:

- P: Variable für die Sprungadressen der Unterprogramme der vier Aufgabentypen
- A$: erster Teil der Formel
- IA: Index des ersten Teils
- B$: zweiter Teil der Formel
- IB: Index des zweiten Teils
- FS: Faktor des gesuchten Stoffes
- FG: Faktor des gegebenen Stoffes
- MV: Zahlenwert der Masse bzw. des Volumens des gegebenen Stoffes
- R: Rechenergebnis des Nutzers

Im Hauptprogramm werden die Felder eingelesen, und die Analyse der Aufgabe entsprechend den vier Typen wird verlangt. Für jeden der vier Aufgabentypen gibt es ein Unterprogramm, in dem das Ergebnis errechnet wird. Dabei wird durch Multiplikation mit 100, INT und Division durch 100 [INT (. . . * 100 +.5)/100] eine Rundung auf zwei Dezimalstellen vorgenommen.
Ein weiteres Unterprogramm verlangt die Eingabe der Formeln und Indizes und ordnet die relativen Molekül- bzw. Atommassen (im folgenden als molare Masse bezeichnet) den entsprechenden Formelteilen zu (Zeilen 550 bis 590). Außerdem wird die Berechnung der molaren Masse für den betreffenden Stoff ausgeführt (Zeile 610). Je nach Stellung der Marke (K = 3 bzw. 23) wird die berechnete molare Masse dem gesuchten bzw. gegebenen Stoff zugeordnet. Ein weiteres Unterprogramm fordert zur Eingabe der Faktoren sowie des Zahlenwertes der gegebenen Größe Masse oder Volumen auf und verarbeitet sie (Zeilen 790 bis 930 und Bild 2).
Die Überprüfung der Richtigkeit des Resultats wird ebenfalls durch ein

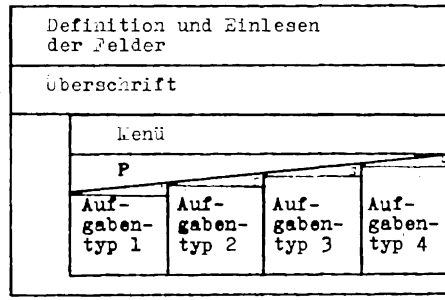
```
CHEMISCHES RECHNEN
xxxxxxxxxxxxxxxxxxxx

NUR MASSEN

FAKTOREN VOR DEN CHEMISCHEN ZEICHEN
DES GESUCHTEN UND DES GEGEBENEN STOF-
FES IN DER REAKTIONSGLEICHUNG
SOWIE DER GEGEBENEN GROESSE

  GIB EIN!

FAKTOR VOR DEM GESUCHTEN STOFF:
       FS = 2

FAKTOR VOR DEM GEGEBENEN STOFF:
       FG =

GEGEBENE MASSE ODER VOLUMEN:

    OHNE EINHEIT!!

    M = ODER V = 47
```

◀ Bild 2. Eingabe der Faktoren (Aufgabentyp 1)

```
CHEMISCHES RECHNEN
xxxxxxxxxxxxxxxxxxxx

GIB DEIN ERGEBNIS
OHNE EINHEIT EIN!

    X = 39.7

RICHTIG! DAS GENAUE
ERGEBNIS IST:

X = 39.98
    EINHEIT DER GEGEBENEN MASSE
```

Bild 3. Ergebnisteil

Unterprogramm vorgenommen. (Zeilen 1600 bis 1690 und Bild 3).
In Bild 4 ist abschließend das Struktogramm für das Hauptprogramm und für das Unterprogramm Aufgabentyp 1 gezeigt. Das komplette BASIC-Programm folgt auf S. 43.



▲ Bild 4. Übersicht über die Programmstruktur

Unterprogramm: 1. Aufgabentyp

M = 3
Unterprogramm: Eingabe der chemischen Zeichen für gesuchten bzw. gegebenen Stoff Berechnen der molaren Masse des gesuchten Stoffes (MS)
M = 23
Unterprogramm: Eingabe der chemischen Zeichen für gesuchten bzw. gegebenen Stoff Berechnen der molaren Masse des gegebenen Stoffes (MG)
Unterprogramm: Eingabe der Faktoren für den gesuchten (FS) und den gegebenen Stoff (FG) sowie der gegebenen Masse bzw. des gegebenen Volumens (MV)
Berechnen des Ergebnisses (E)
Unterprogramm: Eingabe des vom Nutzer errechneten Ergebnisses Vergleich mit dem Ergebnis des Computers Wertung
Ergänzung der Ergebnisausgabe

```
 10 WINDOW 0,31,0,39 : COLOR 7,1 : CLS
 20 !
 30 ! ===== P R O G R A M M S T A R T ===
 40 DIM X$(37),Y(37)
 50 FOR L=0 TO 37 : READ X$(L),Y(L) : NEXT
 60 DATA H,1.01,LI,6.94,C,12,N,14.01,O,16,F,19,NA,22.99,MG,24.31,
    AL,26.98
 70 DATA SI,28.09,P,30.97,S,32.06,CL,35.45,K,39.1,CA,40.08,BR,
    79.91,SN
 80 DATA 118.69,I,126.9,BA,137.34,PB,207.19,CR,52,MN,54.94,FE,
    55.85
 90 DATA CU,63.54,ZN,65.37,AG,107.87,HG,200.59,NH4,18.05,NO3,
    62.01,NO2
100 DATA 46.01,SO4,96.06,SO3,80.08,CO3,60,PO4,94.97,MNO4,118.94,
    CLO3,83.45
110 DATA OH,17.01," ",0
120 !
130 ! ... UEBERSCHRIFT UND MENUE ...
140 !
150 PRINTAT(3,10);"CHEMISCHES RECHNEN"
160 PRINTAT(4,9);STRING$(20,"x")
170 WINDOW 5,31,0,39 : CLS
180 PRINTAT(8,7);"AUFGABENTYPEN"
190 PRINTAT(10,3);"MASSEN GEGEBEN UND GESUCHT (1)"
200 PRINTAT(11,3);"VOLUMEN GEGEBEN UND GESUCHT (2)"
210 PRINTAT(12,3);"VOLUMEN GESUCHT (MASSE GEGEBEN) (3)"
220 PRINTAT(13,3);"VOLUMEN GEGEBEN (MASSE GESUCHT) (4)"
230 PRINTAT(19,3);"ANTWORTE MIT"
240 LOCATE 15,3 : INPUT "1 ODER 2 ODER 3 ODER 4";P
250 ON P GOSUB 980,1110,1250,1410 : GOTO 170
260 !
270 ! === U N T E R P R O G R A M M ====
280 ! ------ EINGABE DER CHEMISCHEN
290 !        ZEICHEN DES GESUCHTEN UND
300 !        DES GEGEBENEN STOFFES -----
310 PRINTAT(8,5);"CHEMISCHE ZEICHEN UND INDIZES"
320 PRINTAT(9,5);"FUER DIE BERECHNUNG"
330 PRINTAT(10,5);"DER MOLAREN MASSEN:"
340 PRINTAT(11,3);"GIB DAS CHEMISCHE ZEICHEN DES"
350 PRINTAT(12,3);"STOFFES EIN, INDEM DU"
360 PRINTAT(13,3);"DIE CHEMISCHEN ZEICHEN DER ELEMENTE"
370 PRINTAT(14,3);"UND SAEURERESTE DEN BUCHSTABEN A"
380 PRINTAT(15,3);"UND B ZUORDNEST   UND DIE INDIZES"
390 PRINTAT(16,3);"DEN BUCHSTABEN IA UND IB."
400 PRINTAT(18,3);"BEACHTE, DASS BEI DEN SAEURERESTEN"
410 PRINTAT(19,3);"NUR DIE ANZAHL DER VOLLSTAENDIGEN"
420 PRINTAT(20,3);"SAEURERESTE EINGEGEBEN WIRD!!"
430 A$=" " : B$=" " : IA=0 : IB=0
440 IF K=3 THEN PRINTAT(23,3);"GESUCHTER STOFF:"
450 IF K=23 THEN PRINTAT(23,22);"GEGEBENER STOFF:"
460 LOCATE 20,K+3 : INPUT "A =";A$
470 LOCATE 21,K+2 : INPUT "IA =";IA
480 LOCATE 22,K+3 : INPUT "B =";B$
490 LOCATE 23,K+2 : INPUT "IB =";IB
500 IF IA=0 THEN IA=1
510 IF IB=0 THEN IB=1
520 !
530 ! ..... AUSWAHL DER KONSTANTEN
540 !        MOLAREN MASSEN .....
550 FOR I=0 TO 37
560 IF A$=X$(I) THEN A=Y(I)  : GOTO 580 : ELSE NEXT I
570 GOSUB 670 : GOTO 460
580 FOR J=0 TO 37
```

```
590 IF B$=X$(J) THEN B=Y(J) : GOTO 610 : ELSE NEXT J
600 GOSUB 670 : GOTO 460
610 Z=0 : Z=A*IA+B*IB
620 IF K=3 THEN MS=Z : ELSE MG=Z
630 RETURN
640 !
650 ! ..... FALSCHES CHEMISCHES
660 !       ZEICHEN ...................
670 WINDOW 24,31,K,K+15 : CLS
680 PRINTAT(25,K);"FALSCHES CHE-"
690 PRINTAT(26,K);"MISCHES ZEICHEN!"
700 PRINTAT(27,K);"EINGABE WIE-"
710 PRINTAT(28,K);"DERHOLEN!"
720 PAUSE 50 : CLS: WINDOW 5,31,0,39
730 RETURN
740 !
750 ! === U N T E R P R O G R A M M ====
760 ! ------ EINGABE DER FAKTOREN
770 !        UND DES GEGEBENEN
780 !        ZAHLENWERTES ------------
790 PRINTAT(8,3);"FAKTOREN VOR DEN CHEMISCHEN ZEICHEN"
800 PRINTAT(9,3);"DES GESUCHTEN UND DES GEGEBENEN STOF-"
810 PRINTAT(10,3);"FES IN DER REAKTIONSGLEICHUNG"
820 PRINTAT(11,3);"SOWIE DER GEGEBENEN GROESSE"
830 PRINTAT(14,5);"GIB EIN!"
840 PRINTAT(16,3);"FAKTOR VOR DEM GESUCHTEN STOFF:"
850 FS=0 : FG=0 : MV=0
860 LOCATE 12,10 : INPUT "FS =";FS
870 PRINTAT(19,3);"FAKTOR VOR DEM GEGEBENEN STOFF:"
880 LOCATE 15,10 : INPUT "FG =";FG
890 PRINTAT(22,3);"GEGEBENE MASSE ODER VOLUMEN:"
900 PRINTAT(24,8);"OHNE EINHEIT!!"
910 LOCATE 22,8 : INPUT "M = ODER V =";MV
920 IF FS=0 THEN FS=1
930 IF FG=0 THEN FG=1
940 CLS : RETURN
950 !
960 ! === U N T E R P R O G R A M M ====
970 ! ------ AUFGABENTYP 1 -------------
980 PRINTAT(6,14);"NUR MASSEN"
990 PRINTAT(7,14);STRING$(10,"-")
1000 K=3 : MS=0 : GOSUB 310
1010 K=23 : MG=0 : GOSUB 430
1020 CLS : GOSUB 790
1030 H=0 : H=INT(FS*MS*MV*100/FG/MG+.5)/100
1040 GOSUB 1600
1050 PRINTAT(19,8);"EINHEIT DER GEGEBENEN MASSE"
1060 PRINTAT(20,4);STRING$(31,"=")
1070 PAUSE 100 : RETURN
1080 !
1090 ! === U N T E R P R O G R A M M ===
1100 ! ------ AUFGABENTYP 2 ------------
1110 PRINTAT(6,14);"NUR VOLUMEN"
1120 PRINTAT(7,14);STRING$(11,"-")
1130 PRINTAT(9,5);"(BEIDE STOFFE UNTER GEGEBENEN"
1140 PRINTAT(10,5);"BEDINGUNGEN GASFOERMIG!)"
1150 PAUSE 50 : WINDOW 8,31,0,39 : CLS : WINDOW 5,31,0,39
1160 GOSUB 790
1170 H=0 : H=INT(FS*MV*100/FG+.5)/100
1180 GOSUB 1600
1190 PRINTAT(19,8);"EINHEIT DES GEGEBENEN VOLUMENS"
1200 PRINTAT(20,4);STRING$(34,"=")
1210 PAUSE 100 : RETURN
```

```
1220 !
1230 ! === U N T E R P R O G R A M M ===
1240 ! ------ AUFGABENTYP 3 ------------
1250 PRINTAT(6,5);"VOLUMEN GESUCHT, MASSE GEGEBEN"
1260 PRINTAT(7,5);STRING$(30,"-")
1270 PRINTAT(9,5);"(GESUCHTER STOFF UNTER DEN GEGE-"
1280 PRINTAT(10,5);"BENEN BEDINGUNGEN GASFOERMIG!)"
1290 PRINTAT(12,5);"!! MASSEN IN GRAMM UMRECHNEN !!" : PAUSE 80
1300 WINDOW 8,31,0,39 : CLS : WINDOW 5,31,0,39
1310 K=23 : MG=0 : GOSUB 310
1320 CLS : GOSUB 790
1330 H=0 : H=INT(FS*MV*2240/FG/MG+.5)/100
1340 GOSUB 1600
1350 PRINTAT(19,8);"LITER"
1360 PRINTAT(20,4);"========="
1370 PAUSE 100 : RETURN
1380 !
1390 ! === U N T E R P R O G R A M M ===
1400 ! ------ AUFGABENTYP 4 ------------
1410 PRINTAT(6,5);"MASSE GESUCHT, VOLUMEN GEGEBEN"
1420 PRINTAT(7,5);STRING$(30,"-")
1430 PRINTAT(9,5);"(GEGEBENER STOFF UNTER DEN GEGE-"
1440 PRINTAT(10,5);"BENEN BEDINGUNGEN GASFOERMIG!)"
1450 PRINTAT(12,5);"!! VOLUMEN IN LITER UMRECHNEN !!" : PAUSE 80
1460 WINDOW 8,31,0,39 : CLS : WINDOW 5,31,0,39
1470 K=3 : MS=0 : GOSUB 310
1480 CLS : GOSUB790
1490 H=0 : H=INT(FS*MS*MV*100/FG/22.4+.5)/100
1500 GOSUB 1600
1510 PRINTAT(19,8);"GRAMM"
1520 PRINTAT(20,4);"========="
1530 PAUSE 100 : RETURN
1540 !
1550 ! === U N T E R P R O G R A M M ===
1560 ! ------ ERGEBNIS DES NUTZERS
1570 !        VERGLEICH MIT
1580 !        RECHNERERGEBNIS
1590 !        WERTUNG ------------------
1600 PRINTAT(10,4);"GIB DEIN ERGEBNIS"
1610 PRINTAT(11,4);"OHNE EINHEIT EIN!"
1620 LOCATE 8,8 : INPUT "X ="$R
1630 IF R>H+H/100 OR R<H-H/100 THEN 1640 : ELSE 1670
1640 PRINTAT(15,10);"FALSCH!"
1650 PAUSE 50 : WINDOW 13,15,0,39 : CLS : WINDOW 5,31,0,39
1660 GOTO 1620
1670 PRINTAT(15,4);"RICHTIG! DAS GENAUE"
1680 PRINTAT(16,4);"ERGEBNIS IST:"
1690 PRINTAT(18,4);"X =";H
1700 RETURN
```

Mögliche Weiterführungen

Seit Einführung des Taschenrechners in den Schulen kann eine höhere Genauigkeit der Schülerrechnungen verlangt werden. Die Einschränkung in Zeile 1630 könnte also entfallen. Wegen der Zuverlässigkeitsbetrachtungen der Ergebnisse ist es aber wenig sinnvoll, die Konstanten in den DATA-Listen noch genauer anzugeben.

Zum Austesten des Programms vor dem Einsatz bei Schülern kann eingefügt werden:

```
1615 WINDOW15, 26, 30, 39 : ? H :
     WINDOW5, 31, 0, 39
```

Es ist dann nicht nötig, die betreffende Aufgabe erst mit dem Taschenrechner

auszurechnen, bevor das Ergebnis durch den anleitenden Lehrer in den Computer eingegeben wird.
An einigen Stellen des Programms sind technisch bzw. didaktisch sinnvolle Ergänzungen nötig. Die Eingaben der Indizes für die Formeln des gesuchten und gegebenen Stoffes können gegen sinnlose Zahlenwerte geschützt werden ($1 \leq I \leq 8$).
Didaktisch sinnvoll ist auch eine Begrenzung der Anzahl wiederholter Falscheingaben der Lösung, z. B. auf zwei.
Aufwendiger ist eine Wiedergabe aller Eingaben im Unterprogramm für die Faktoren der gesuchten und gegebenen Stoffe. Sie könnte dem Nutzer zum Vergleich mit seiner schriftlich vorliegenden chemischen Gleichung und den Forderungen der Aufgabe dienen.

Für einige Hinweise danke ich Herrn Dr. LÖHR.

Autor:
StR *Klaus Bockhacker*
Pädagogischer Mitarbeiter
Haus der Pioniere „Erich Weinert"
Jena

Der Bildungscomputer robotron A 5105

Der Bildungscomputer (BIC) A 5105 setzt die Kleincomputerlinie KC 85/3 und KC 87 auf einem wesentlich höheren Niveau fort, wobei eine Softwarekompatibilität zum PC 1715 gesichert wurde. Durch den Einsatz modernster Schaltkreise und rationellster Fertigungstechnologien ist eine Produktion von sehr hohen Stückzahlen möglich.
Der BIC besteht aus drei Baugruppen:

Computergrundgerät (CGG) mit
- Computerflachtastatur (8-Bit-Zeichensatz)
- Mikroprozessor UA 880 D mit einer Taktfrequenz von 3,75 MHz
- 48 (oder 64) kByte ROM als Programmspeicher für das Basisbetriebssystem RBASIC
- 64 kByte RAM als Arbeitsspeicher
- 128 kByte RAM als Bildspeicher
- Videocontroller U 82720
- Kassetteninterface

Diskettenspeichereinheit (DSE) mit
- 1 Diskettenlaufwerk 1.6 (5 1/4 Zoll)
- Floppy-Disk-Controller U 8272 (max. 3 Laufwerke, 2 Laufwerke im Beistellgerät)
- V.24-Druckerinterface
- V.24-Plotterinterface
- 2 × 8-Bit-Parallel-Interface für Schülerexperimentiergerät
- Netzanschluß für ROLANET-Rechnernetze

Monitor (MON) mit
- monochromer 12-Zoll-Bildröhre und/oder Anschluß eines Farbmonitors/Farbfernsehers mit RGB-Eingang:

  alphanumerische Ausgabe

  25 Zeilen × 40 oder 80 Zeichen,
  16 Vordergrund-/8 Hintergrundfarben

  grafische Ausgabe

  320 × 200 Pixel mit 16 Farben
  640 × 200 Pixel mit 4 aus 16 Farben

Neben dem ROM-orientierten Basisbetriebssystem RBASIC ist das Betriebssystem SCPX 5105 möglich, das vollständig zum Betriebssystem SCPX 1715 kompatibel ist.

Spielprogrammierung

– Von der Idee zum fertigen Spiel –

(Teil 2)

Schiebefax

Andere Namen sind taquin, Boß-Puzzle und 15er Spiel. In einem quadratischen flachen Kasten liegen 15 durchnumerierte Spielsteine, die unter Ausnutzung des 16., freien Platzes in eine geordnete Reihenfolge zu schieben sind (Bild 1).

Dieses Spiel wurde 1878 oder nach einer anderen Quelle Anfang der 70er Jahre des 19. Jahrhunderts von SAMUEL LLOYD erfunden. 1879 veröffentlichten W. JOHNSON und W. E. STORY die Theorie dazu in »American Journal of Mathematics«. Diese besagt, daß nur die geraden Permutationen eine lösbare Aufgabe darstellen. Gerade Permutationen sind solche Anordnungen der Zahlenplättchen, die durch eine gerade Anzahl von Vertauschungen je zweier Plättchen aus der geordneten Aufstellung hervorgehen. Eine besonders ausführliche Darstellung der Theorie enthält [6].

Obiges Bild kann z. B. entstanden sein durch Vertauschen von 1 mit 5, danach mit 9, 11, 4, 7 und 6 sowie von 3 mit 15 und von 8 mit 14, 10 und 12; das sind insgesamt 10 Vertauschungen. Da 10 eine gerade Zahl ist, ist diese Aufgabe also lösbar.

1. Lösungsalgorithmus

Die Plättchen werden in der Reihenfolge

1 2 3 4 5 6 7 8
9 13 10 14 11 15 12

an die jeweils für sie vorgesehenen Plätze gebracht, wobei in der Regel alle jeweils bereits geordneten Plättchen ungestört bleiben müssen. In Ausnahmefällen dürfen vorangehende Plättchen für kurze Zeit ihren bereits erreichten Platz verlassen, um das Einordnen weiterer Plättchen damit zu ermöglichen. Solche Ausnahmen ergeben sich bei der Behandlung der 4. Spalte und der 4. Zeile.

Es ist zweckmäßig, das Programm stückweise aufzubauen und jede Etappe gesondert zu testen. Fehler beim Nach-

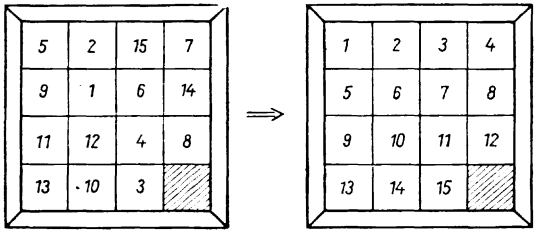


Bild 1

programmieren werden dadurch leichter erkannt, die Verfahrensweise in der Programmniederschrift wird zugleich verständlicher. Konkrete Hinweise zum Lösungsalgorithmus sind dazu in die Programmeinführung an entsprechender Stelle eingefügt.

2. Programmeinführung

Die **erste** Etappe ist der Aufbau des Spielfeldes auf dem Bildschirm. Dazu sind zunächst die Programmzeilen 20, 50, 270 bis 320 und 440 bis 540 erforderlich. Wenn diese Zeilen richtig eingegeben wurden, erscheint nach Programmstart (RUN) auf dem Bildschirm das geordnete Spiel. Zeile 480 erzeugt unterschiedliche Farben bzw. Grautöne bei Schwarz-Weiß-Empfängern jeweils für Plättchen mit gerader oder ungerader Beschriftung.
Erst nach Erprobung dieses Teils sollte die Eingabe des Programms fortgesetzt werden.
Die **zweite** Etappe besteht im Durcheinanderbringen der Plättchen. Dies geschieht durch die Programmzeilen 340 bis 420.
Die Idee ist hierbei folgende:
Um eine gerade Permutation zu erzeugen, werden nacheinander für das erste bis 14. Plättchen mit einem Zufallsgenerator Tauschpartner bestimmt. An diese Tauschpartner sind dabei die Bedingungen zu stellen, daß

- kein Plättchen mit sich selbst vertauscht werden darf (dafür sorgt Programmzeile 380) und
- kein Plättchen mit dem Leerfeld vertauscht wird (Programmzeile 390).

Anderenfalls wäre die Permutation nicht zwangsläufig gerade. Nachdem auch hierfür die Wirksamkeit durch einen Programmstart nachgewiesen wurde, geht es an die **dritte** Etappe. Hierin sollen die Plättchen beweglich gemacht werden, zunächst erst einmal mittels Handsteuerung. Es sind die Zeilen 560 bis 590, 620 bis 660, 710 bis 770 sowie

```
680 GOTO 580
780 GOTO 580
```

einzugeben und das Programm erneut zu starten. Wenn alles richtig eingegeben wurde, müssen nunmehr alle Zahlenfelder auf die Cursorsteuertasten reagieren. Für das Programmverständnis ist hierbei zu beachten, daß die Cursorsteuertasten die Bewegung von Zahlenfeldern auf das Leerfeld auslösen sollen, intern im Rechner aber das Leerfeld als beweglich angesehen wird, die Zahlenfelder somit passiv zu sein scheinen. Konkret wirkt sich dies in einer Umkehr der Bewegungsrichtung aus, so wie sie in den Programmzeilen 620 bis 650 realisiert ist.
Die Zeilen 590 und 660 tragen dafür Sorge, daß die Betätigung falscher Tasten ohne Wirkung bleibt, insbesondere verhindern die Zusatzbedingungen in den Zeilen 620 bis 650 eine Überschreitung des Spielfeldrandes. Das so gestaltete Spiel läuft endlos, es muß deshalb als nächstes um eine Endeabfrage ergänzt werden.
Vierte Etappe – Endeabfrage. Hierzu werden die Programmzeilen 250, 260 und 860 bis 950 eingegeben sowie die Zeile 780 gelöscht (Testzeile). Die Zeile 250 wird zunächst nur als Ansprungstelle von Zeile 940 benötigt, später gehört sie zum Abschluß der Bedienungsanleitung. In den Zeilen 860 bis 890 erfolgt das Durchsuchen des Spielfeldes nach dem ersten, in der natürlichen Reihenfolge nicht richtig positionierten Plättchen, welches zum Rücksprung zur Eingabe führt.
Erst jetzt, in der **fünften** Etappe kann mit der automatischen Bewegungssteuerung begonnen werden. Für die Bedienung des später laufenden Programms werden die Zeilen 550, 600

und 610 eingesetzt und weiterhin die Zeilen 960 bis 1110 angefügt. Zur Testung dienen die beiden Zeilen

1130 PRINT AT (28,5); N; Z; S
1140 GOTO 550

Wenn im Programmlauf der Buchstabe A oder der Buchstabe H gedrückt werden, so erscheinen unterhalb des Spielfeldes die Nummer des im Moment zu bearbeitenden Plättchens und dessen Position als Zeilen- und Spaltennummer. Später sollen „H" das Einrichten eines Plättchens und „A" einen vollständig automatischen Ablauf auslösen.

Etappe **sechs** besteht im Heranholen des Leerfeldes an das aktuelle Plättchen. Das hat den Zweck, für die folgenden Schritte die Anzahl der unterschiedlichen Fälle erheblich zu reduzieren. Zeile 1140 wird gelöscht, neu hinzugenommen werden die Zeilen 1310 bis 1420. Weiterhin ist zu beachten, daß die automatische Bewegung einzelne Programmteile der Handsteuerung mit nutzt. In B$ wird die zutreffende Cursorsteuerung nachgebildet und anschließend in Zeile 590 mit der Analyse der Eingabe fortgesetzt. Damit die automatische Bewegung nicht ins Stocken gerät, erfolgt aus dem Handsteuerteil an geeigneter Stelle der Rücksprung, d.h., es sind die Zeilen 680 und 690 sowie 780 und 840 einzufügen. Zum besseren Testen ist noch folgende Zeile erforderlich:

1440 GOTO 550

Der Test besteht nun darin, mittels Handsteuerung das Leerfeld möglichst weit vom aktuellen Plättchen zu entfernen und anschließend mit den Tasten A und H die Automatik anzusprechen. Das Programm arbeitet richtig, wenn das Leerfeld ohne Zerstörung bisher geordneter Felder auf dem kürzest möglichen Weg an das aktuelle Feld herangeführt wird, also gerade oder schräg benachbart ist.

Getestet werden muß auch, was dann passiert, wenn diese Nachbarschaft von vornherein gegeben ist. Es darf nichts passieren! Die Notwendigkeit letztgenannter Tests wird oft unterschätzt. Ist das Programm aber erst einmal fertig, dann lassen sich aus solchen Unterlassungen herrührende Fehler nur noch sehr schwer orten. Der Aufwand, um solche Fehler zu beseitigen, kommt mitunter dem einer Neuprogrammierung gleich.

Die **siebente** Etappe hat vorbereitenden Charakter und kann nicht als selbständiges Programm angesehen werden. Zur Realisierung komplexerer Bewegungsabläufe werden sechs Unterprogramme (UP) eingeführt. Sie befinden sich auf den Programmzeilen 1650 bis 1760 und 2010 bis 2410. Außerdem ist die Zeile 1440 zu ändern in

1440 GOTO 560

Der Test erfolgt hierbei durch direktes Ansprechen des jeweiligen Unterprogramms an seiner Startadresse. Zu beachten ist jedoch, daß als Vorleistung ein normaler Programmstart und Starten der Automatik erforderlich sind, also eine Nutzung der bisherigen Programmteile, um in allen Arbeitsbereichen einen geeigneten Ausgangszustand zu erzeugen. Nach einem erfolgten Programmabbruch darf das ausgewählte UP aber nur mit GOTO gestartet werden, da bei Start mit RUN und Adresse alle Arbeitsbereiche wieder gelöscht werden würden.

Somit ergibt sich folgender Testablauf:

- RUN
- Ein paar Cursorsteuertasten zur Überprüfung der Beweglichkeit bzw. zur Erzielung einer geeigneten Ausgangsstellung
- A
- BRK-Taste
- GOTO 1660 bzw. GOTO und die Startadresse eines anderen sinnvollen UP.

(Sinnlos sind in diesem Fall Versuche, den Rand zu überschreiten, da zum Abfangen solcher Schnitzer nicht die neu eingefügten, sondern bereits ausgeteste Programmteile verantwortlich sind.)

Der oder die letzten Schritte sind so oft zu wiederholen, bis eine ausreichende Sicherheit im Umgang mit diesen Unterprogrammen gewährleistet ist. Um den Ablauf ordentlich verfolgen zu können, ist eine „Bremse" einzubauen. Dazu dienen die Programmzeilen 800 bis 830. Die obere Grenze in der Zeile 800 bestimmt das Tempo, welches individuell festzulegen ist. Bei der Entwicklung der Unterprogramme hat sich darüber hinaus das Legen einer Spur bewährt. Das bedeutet, daß die Unterprogramme mit Bildschirmausschriften beginnen, die unmittelbar am UP-Ende wieder gelöscht werden. Dadurch kann man Fehler leichter lokalisieren. Eine Spur kann z.B. so aussehen:

```
1345 PRINT AT (29,5); "LEER RAN"
1440 PRINT AT (29,5); "          "
1450 GOTO 450
2020 PRINT AT (29,5); "HEBEN"
2025 IF S=L AND K+1=Z THEN 1940
2265 PRINT AT (29,5); "     "
2295 PRINT AT (29,5); "SEITE"; D
2305 PRINT AT (29,5); "     "
```

Zum Aufbau der beiden Unterprogramme zum Heben und zur seitlichen Bewegung ist folgendes zu berücksichtigen:

- Wo befindet sich das aktuelle Plättchen?
  Spalte S und Zeile Z, belegt in den Programmzeilen 1090 und 1100.
- Wohin soll das aktuelle Plättchen?
  Seine Nummer ist N, gemäß Zeile 1070; es soll auf Zeile IZ und Spalte JZ, welche in den Programmzeilen 1310 und 1320 ermittelt werden. (Wegen des Einmischens der 4. Zeile des Spielfeldes in die dritte, welches wir erst noch behandeln müssen, sind IZ und JZ in der Zeile 1070 noch nicht endgültig festgelegt. N kann noch nachträglich verändert werden müssen.)
- Wo befindet sich zur Zeit das Leerfeld?
  Laut Programmzeile 530 in Zeile K und Spalte L.

Zur Feststellung der konkret anliegenden Aufgabe sind über diese 7 Variablen Falluntersuchungen anzustellen und auf möglichst wenige charakteristische Fälle zurückzuführen. Die Reduzierung der tatsächlichen Bewegung auf jeweils einen UP-Aufruf ist bei der Analyse der beiden oben genannten Unterprogramme sicher eine große Hilfe. Wichtig ist in jedem Fall, daß der Einfluß des Randes sowie bereits geordneter Teile des Spielfeldes ebenfalls beachtet werden, um zu einer einwandfreien Lösung zu kommen.

In der **achten** Etappe sollen die ersten drei Zahlen der ersten beiden Zeilen ihren Platz bereits selber finden, nachdem jeweils die Taste H gedrückt wurde. Hierfür sind die Programmzeilen 1440 bis 1530 zuständig. Nach der gründlichen Vorbereitung in der 7. Etappe beschränken sich nunmehr alle Überlegungen auf die Bewegung des aktuellen Plättchens, welches höchstens drei Mal anzuheben und höchstens drei Mal zur Seite zu schieben ist. Die Reihenfolge dieser beiden Bewegungsarten ist dabei nicht beliebig, bisher Geordnetes soll ja schließlich in seiner Ordnung erhalten bleiben.

Die **neunte** Etappe behandelt die Spalte 4 der ersten beiden Zeilen. Da ein Einrücken ohne Störung der vorangegangenen Zahlen nicht möglich ist, erfordert dies eine Sonderbehandlung. Die Sonderbehandlung wird dort beendet, wo der Rechner mit den bisherigen Programmteilen allein weiterkommt. Es sind die Programmzeilen 1550 bis 1640 einzugeben.

Nach dem Start und der Taste A muß der Rechner mindestens die ersten 9 Plättchen richtig einordnen. Ein

darauf folgender irregulärer Ablauf kann durch die Taste S unterbrochen werden, welche bei der Eingabe der „Bremse" wirksam gemacht wurde.
In der **zehnten** Etappe wird die Programmlogik vervollständigt um die Behandlung der letzten beiden Zeilen. Die Programmzeilen 1130 bis 1290 und 1770 bis 2000 ergeben dies. Die Bewegungsabläufe sind dabei etwas komplizierter, aber viele Vorarbeiten dazu wurden bereits erledigt.
Im Test ist nunmehr das volle Wechselspiel aus Handsteuerung, Hilfe (H), Automatik (A) und Stoppen der Hilfe oder Automatik mit Rückkehr zur Handsteuerung (S) verwendbar. Nach erfolgreichem Test sind die Hilfsausschriften aus der siebenten Etappe wieder zu entfernen, also die Programmzeilen 1345, 2025, 2265, 2295 und 2305 sind zu löschen, und die Programmzeile 2020 muß nunmehr ordentlich eingegeben werden.
Die elfte und letzte Etappe dient der Vervollständigung des Programms. Die Programmzeilen 60 bis 230 erzeugen zum Programmstart eine Bedienungsanleitung auf dem Bildschirm, Zeilen 2420 bis 2440 sorgen für ein ordentliches Programmende und überall einzustreuende Kommentare – soweit noch nicht geschehen – ermöglichen zu einem späteren Zeitpunkt, die Programmlogik leichter zu überblicken und Stellen für gewünschte Änderungen bequemer zu lokalisieren.
Die allerletzte Aufgabe ist nun nur noch die Sicherung des fertigen Programms (s. S. 52) auf einem Magnetband.

Barrikade

Bei diesem Spiel wird ein Spielstein mittels der Cursorsteuertasten auf dem Bildschirm in Einzelschritten bewegt. Nach jedem Schritt baut der Rechner willkürlich eine der noch freien Bewegungsrichtungen zu, errichtet eine Barrikade. Es kommt nun darauf an, den Spielstein so lange wie möglich zu bewegen.

1. Lösung

So einfach, wie es zunächst erscheint, ist diese Aufgabe gar nicht zu realisieren, denn bei den Kleincomputern reicht in der Regel die Speicherkapazität nicht aus, um alle Plätze auf dem Bildschirm intern als Matrix abzubilden. Es muß demzufolge ein direkter Zugriff zum Bildwiederholspeicher erfolgen, um die bisher gesetzten Barrikaden zu erkennen. Zur Schaffung einer Übersicht über die Speicheraufteilung beim Bildwiederholspeicher ist folgendes kleines Programm nützlich:

```
10 WINDOW 0,31,0,39:CLS
20 FOR I=0 TO 10239
30 VPOKE I,255
40 NEXT I
```

Nacheinander wird der gesamte Bildschirm beschrieben. Da dies aber wohl doch etwas zu schnell geht, ist das Programm besser wie folgt abzuändern:

```
10 WINDOW 0,31,0,39:CLS
20 FOR I=0 TO 10239
30 VPOKE I,126
40 PAUSE 1
50 NEXT I
```

Jeweils 8 Bildzeilen bilden ein Zeichen, jeweils 16 Bildzeilen bzw. zwei Zeichenreihen werden gemeinsam erzeugt im Rhythmus

1 5 9 13 2 6 10 14
3 7 11 15 4 8 12 16.

Dies gilt aber nur für die Spalten 0 bis 31. Anschließend werden aus den restlichen Spalten weitere solcher Doppelzeilen gebildet, indem jeweils 4 Zeilenpaare hintereinander gelegt werden. Bei der Programmierung ist somit zu unterscheiden, ob ein Platz

```
10 REM SCHIEBEFAX
20 REM IDEE UND PROGRAMM:WALTER GOERGENS
30 DIM A(4,4) :REM SPIELFELD
40 REM BEDIENUNGSANLEITUNG
50 WINDOW:COLOR 7,1:CLS
60 PRINT:PRINT:PRINT:PRINT
70 PRINT"          DAS BEKANNTE SCHIEBEFAX"
80 PRINT:PRINT
90 PRINT:PRINT"  DIE EINZELNEN SCHEIBEN SIND ZU"
100 PRINT "  ORDNEN, ZEILENWEISE, AUFSTEIGEND."
110 PRINT "  DAZU SIND DIE KURSORTASTEN ZU"
120 PRINT "  BENUTZEN. SIE KOENNEN AUCH DRUECKEN:"
130 PRINT:PRINT INK 6;"  H ";
140 PRINT INK 7;"= HILFE - DER RECHNER ZEIGT,WIE DAS"
150 PRINT"               PLAETTCHEN ZU ZIEHEN HAT."
160 PRINT:PRINT INK 6;"  A ";
170 PRINT INK 7;"= AUTOMATIK - ALLES LAEUFT ALLEINE."
180 PRINT:PRINT INK 6;"  S ";
190 PRINT INK 7;"= STOP -  UEBERGANG VON H ODER A"
200 PRINT "                AUF HANDSTEUERUNG"
210 PRINT:PRINT:PRINT:PRINT:PRINT:PRINT
220 PRINT INK 23;"                                    >ENTER<"
230 A$=INKEY$:IF A$="" THEN 230
240 REM GRUNDSTELLUNG
250 CLS
260 Q=0
270 WINDOW 5,25,10,30:PAPER 6:CLS
280 WINDOW 6,24,11,29:PAPER 0:CLS
290 FOR I=1 TO 4
300 FOR J=1 TO 4
310 A(I,J)=4*I+J-4
320 NEXT J,I
330 REM DURCHEINANDER - ABER NUR GERADE PERMUTATION
340 FOR I=1 TO 4
350 FOR J=1 TO 4
360 K=1+INT(4*RND(1))
370 L=1+INT(4*RND(1))
380 IF I=K AND J=L THEN 360
390 IF K=4 AND L=4 THEN 360
400 IF I=4 AND J=3 THEN 440
410 Z=A(I,J):A(I,J)=A(K,L):A(K,L)=Z
420 NEXT J,I
430 REM ANZEIGE DER ANFANGSSTELLUNG
440 FOR I=1 TO 4
450 FOR J=1 TO 4
460 B=A(I,J)
470 IF B=16 THEN 530
480 PAPER(B-INT(B/2)*2)*2+2
490 WINDOW 5*I+1,5*I+4,5*J+6,5*J+9
500 CLS
510 PRINT AT(5*I+3,5*J+6);B
520 GOTO 540
530 K=I:L=J :REM LEERFELD-POSITION
540 NEXT J,I
550 H=0 :REM KEIN HILFEERSUCHEN
560 WINDOW 5*K+1,5*K+4,5*L+6,5*L+9
570 REM EINGABE SPIELSTEUERUNG
580 B$=INKEY$:IF B$="" THEN 580
590 I=K:J=L :REM SPIELSTEUERUNG
600 IF B$="A" THEN 1000
610 IF B$="H" THEN 970
620 IF B$=CHR$(9) AND L>1 THEN L=L-1
630 IF B$=CHR$(8) AND L<4 THEN L=L+1
640 IF B$=CHR$(11) AND K<4 THEN K=K+1
```

```
650 IF B$=CHR$(10) AND K>1 THEN K=K-1
660 IF I+J<>K+L THEN 710
670 REM BEWEGUNG UNMOEGLICH(RAND ODER UNZUL.EINGABE)
680 IF H=0 THEN 580
690 RETURN
700 REM ZUGAUSFUEHRUNG
710 Q=Q+1
720 B=A(K,L)
730 PAPER(B-INT(B/2)*2)*2+2
740 CLS:PRINT AT(5*I+3,5*J+6);B
750 A(I,J)=A(K,L):A(K,L)=16
760 WINDOW 5*K+1,5*K+4,5*L+6,5*L+9
770 PAPER 0:CLS
780 IF H=0 THEN 860
790 REM VERZOEGERUNG FUER AUTOMATIK MIT UNTERBRECHUNGSABFRAGE
800 FOR X=1 TO 40
810 B$=INKEY$
820 IF B$="S" THEN H=0
830 NEXT X
840 RETURN
850 REM ENDEABFRAGE
860 FOR I=1 TO 4
870 FOR J=1 TO 4
880 IF A(I,J)<>4*I+J-4 THEN 580
890 NEXT J,I
900 PRINT AT(28,5);"MIT";Q;"SCHRITTEN GELOEST !"
910 PRINT AT(29,5);"      NOCH EINMAL (J/N) ?"
920 WINDOW:PAPER 1
930 B$=INKEY$:
940 IF B$="J" THEN 250
950 IF B$<>"N" THEN 930
960 GOTO 2420
970 H=2 :REM BEGINN HILFE
980 GOTO 1020
990 IF H=2 THEN 550
1000 H=1 :REM BEGIN AUTOMATIK
1010 REM WO STEHT DAS ERSTE NICHT RICHTIG EINGEORDNETE PLAETTCHEN
1020 N=0
1030 FOR I=1 TO 4
1040 FOR J=1 TO 4
1050 IF N>0 THEN 1080
1060 IF A(I,J)=4*I+J-4 THEN 1110
1070 N=4*I+J-4
1080 IF A(I,J)<>N THEN 1110
1090 S=J
1100 Z=I
1110 NEXT J,I
1120 REM STRATEGIEAUSWAHL
1130 IF N<10 OR N>12 THEN 1260
1140 REM EINMISCHEN DER LETZTEN IN DIE VORLETZTE ZEILE
1150 IF A(4,1)=13 THEN 1180
1160 N=13
1170 GOTO 1030
1180 IF N<11 THEN 1260
1190 IF A(4,2)=14 THEN 1220
1200 N=14
1210 GOTO 1030
1220 IF N<12 THEN 1260
1230 GOSUB 1720
1240 GOTO 900
1250 REM HIER GEHT'S LOS
1260 IF N<15 THEN 1310
1270 REM SONDERBEHANDLUNG 15
1280 GOSUB 1660
```

```
1290 GOTO 900
1300 REM POSITION ZIELFELD
1310 IZ=INT((N+3)/4)
1320 JZ=N-4*IZ+4
1330 REM LEERFELD HERANHOLEN
1340 U=(L-S)*(L-S)+(Z-K)*(Z-K)
1350 IF U<3 THEN 1440
1360 B$=CHR$(10)
1370 IF S<L-1 THEN B$=CHR$(9)
1380 IF S>L+1 THEN B$=CHR$(8)
1390 IF K<Z-1 THEN B$=CHR$(11)
1400 IF K<Z AND S<JZ THEN B$=CHR$(11)
1410 GOSUB 590
1420 GOTO 1340
1430 REM ZEILE 1 BIS 3
1440 IF N>12 THEN 1770
1450 IF Z=4 THEN GOSUB 2020
1460 IF JZ=S THEN 1490
1470 GOSUB 2290
1480 GOTO 1460
1490 IF IZ=Z THEN 990
1500 IF IZ<Z-1 THEN GOSUB 2020
1510 IF JZ=4 THEN 1550
1520 GOSUB 2020
1530 GOTO 990
1540 REM SONDERZUEGE SPALTE 4
1550 IF Z=K THEN 1590
1560 IF K=IZ THEN 1630
1570 IF L=4 THEN GOSUB 1690
1580 GOSUB 1750
1590 GOSUB 1690
1600 GOSUB 1750
1610 GOSUB 1660
1620 GOSUB 1660
1630 GOSUB 1720
1640 GOTO 990
1650 REM UP "LINKS"
1660 B$=CHR$(8)
1670 GOTO 590
1680 REM UP "RECHTS"
1690 B$=CHR$(9)
1700 GOTO 590
1710 REM UP "HINAUF"
1720 B$=CHR$(11)
1730 GOTO 590
1740 REM UP "HERUNTER"
1750 B$=CHR$(10)
1760 GOTO 590
1770 REM LETZTE ZEILE
1780 IF S>JZ+2 THEN GOSUB 2290
1790 IF S>JZ+1 THEN GOSUB 2290
1800 IF L=JZ THEN GOSUB 1660
1810 IF A(IZ,JZ)=N THEN 990
1820 IF L=JZ+1 THEN 1860
1830 IF K=3 THEN GOSUB 1720
1840 IF Z=4 THEN 1890
1850 GOSUB 1690
1860 IF K=4 THEN GOSUB 1750
1870 GOSUB 1660
1880 GOSUB 1720
1890 GOSUB 1690
1900 GOSUB 1690
1910 GOSUB 1750
1920 GOSUB 1660
```

```
1930 GOSUB 1720
1940 GOSUB 1660
1950 GOSUB 1750
1960 GOSUB 1690
1970 GOSUB 1690
1980 GOSUB 1720
1990 GOSUB 1660
2000 GOTO 990
2010 REM UP "AUSGEWAEHLTES PLAETTCHEN UM EINE ZEILE HEBEN"
2020 IF S=L AND K+1=Z THEN 2250
2030 IF Z<4 THEN 2070
2040 IF U=1 THEN GOSUB 1750
2050 ON 2+L-S GOSUB 1660,1660,1690
2060 GOTO 2250
2070 IF K<Z THEN 2050
2080 IF S=4 THEN 2210
2090 IF L=S THEN 2160
2100 IF L<S THEN 2130
2110 IF K>Z THEN 2170
2120 GOTO 2180
2130 IF Z>IZ+1 THEN 2220
2140 IF K=Z THEN GOSUB 1720
2150 GOSUB 1660
2160 GOSUB 1660
2170 GOSUB 1750
2180 GOSUB 1750
2190 GOSUB 1690
2200 GOTO 2250
2210 IF S=L THEN GOSUB 1690
2220 IF K>Z THEN GOSUB 1750
2230 GOSUB 1750
2240 GOSUB 1660
2250 GOSUB 1720
2260 Z=Z-1
2270 RETURN
2280 REM UP "AUSGEWAHLTES PLAETTCHEN EIN FELD ZUR SEITE"
2290 D=(JZ-S)/ABS(JZ-S)+S
2300 IF K<>Z THEN 2340
2310 IF L=D THEN 2390
2320 IF Z=4 THEN GOSUB 1750
2330 IF Z<4 THEN GOSUB 1720
2340 IF L=D THEN 2380
2350 ON 2+S-D GOSUB 1660,1660,1690
2360 IF L=D THEN 2380
2370 ON 2+S-D GOSUB 1660,1660,1690
2380 ON 2+K-Z GOSUB 1720,1750,1750
2390 ON 2+D-S GOSUB 1660,1690,1690
2400 S=D
2410 RETURN
2420 REM PROGRAMMENDE
2430 WINDOW
2440 PAPER 1
```

unter den ersten 32 Spalten gewählt wird oder einer dahinter. Für die oberste Bildzeile eines Zeichens in Zeile Z und Spalte S gilt somit für

S<32 und Z gerade:
A = Z * 256 + S
Z ungerade:
A = Z * 256 + S — 192

S>31 und Z gerade:
A = Z * 4 + INT (Z/8) * 480 + + S + 8160
Z ungerade:
A = Z * 4 + INT (Z/8) * 480 + + S + 8220

Alle Bildschirmzeilen werden erreicht durch Zuschläge zu diesen Werten von 0 128 256 384 32 160 288 416.

Die Umrechnung von S und Z in die zugehörige Adresse im PIXEL-RAM, wie der Bildwiederholspeicher auch genannt wird, erfolgt am zweckmäßigsten in einem Unterprogramm. Zum einen erleichtert dies eine Nachnutzung in anderen Programmen und zum anderen ist dann bei der Bearbeitung des eigentlichen Programms diese Teilaufgabe bereits „um die Ecke".
Zur Erprobung dieses Unterprogramms, welches aus den Zeilen 490 bis 550 des auf S. 57 dargestellten Programms zu entnehmen ist, benötigt man einen Testrahmen. Im Beispiel soll ein beliebiges Zeichen oben darüber einen Querstrich erhalten:

```
10 WINDOW 0,31,0,39:CLS
20 INPUT"SPALTE,ZEILE:";S,Z
30 INPUT"ZEICHEN:";A$
40 PRINT AT(Z,S);A$
50 GOSUB 500
60 VPOKE A,254
70 GOTO 20
```

Zum besseren Verständnis sei gesagt, daß die Zahl in der Zeile 60 als Dualzahl aufgefaßt werden muß, um zu den konkreten Bildpunkten zu gelangen. Also 254 = 11111110 (bei Zahlenbasis 2).

2. Programmeinführung

Die Programmzeilen 20 bis 70 dienen der Nutzerinformation und haben für die Verwirklichung der Spielidee an sich keine Bedeutung. Sie erleichtern jedoch anderen Nutzern den Umgang mit diesem Programm.
Durch eine Randbegrenzung (Zeilen 80 bis 130) erreicht man, daß im späteren Verlauf der Programmabarbeitung keine spezielle Abfrage einer eventuellen Randüberschreitung mehr erforderlich ist. Der Rand wirkt dann intern wie eine bereits errichtete Barrikade. Als Besonderheit ist hinzuzufügen, daß eine zeitweilige Umschaltung in den PAGE-Modus erforderlich ist, damit das Bild nicht beim Beschreiben des unteren Randes wegläuft.
Die Zeilen 140 und 150 stellen alle zulässigen Bewegungsrichtungen des Spielsteines dar, welcher in den Zeilen 160 bis 180 willkürlich im Spielfeld eingesetzt wird. Danach folgt der Hauptzyklus, welcher das eigentliche Spielen ausmacht. Zuvor ist es aber erforderlich, den bisher erarbeiteten Teil einem gründlichen Test zu unterziehen. Das geht durch Ergänzung der bisher angeführten Programmzeilen um

```
190 END
```

Lustig sieht es auch aus, wenn statt dessen eingesetzt wird

```
190 GOTO 160
```

Wer will, kann für letzteres auch ein anderes Zeichen, mittels RND variierte Farben usw. verwenden.
Bei der Behandlung der manuellen Spielsteinführung ist besondere Sorgfalt auf die Verhinderung von Auswirkungen durch Bedienfehler zu legen. Es sind deshalb Kontrollen über die Zulässigkeit der betätigten Tasten (Zeilen 210 und 220) und über die Ausführbarkeit des damit beabsichtigten Zuges unbedingt erforderlich, um das Spiel später reibungslos in der beabsichtigten Weise ablaufen lassen zu können. Die zuletzt genannte Überprüfung besteht aus den Teilschritten

- Merken der bisherigen Position (Zeile 230),
- Berechnen des Zielfeldes (Zeilen 240 bis 270),
- Umrechnen des Zielfeldes in die PIXEL-RAM-Adresse und
- Lesen im PIXEL-RAM zur Abfrage der Belegung (Zeilen 280 bis 300).

Nun kann gezogen werden (Zeilen 310 bis 330). Auch hier sollte ein Test des bisher erarbeiteten Teils eingefügt werden. Durch

```
340 GOTO 200
```

entsteht ein brauchbarer Zyklus. Bei richtiger Eingabe der Zeilen 190 bis 330 und der Testhilfe in Zeile 340 kann der Stein auf dem ganzen Bildschirm frei bewegt werden und wird vom Rand sicher aufgehalten.

Der Rechnerzug besteht nun darin, eine beliebige, noch freie Bewegungsrichtung zu blockieren. Die Auswahl aus allen noch freien Richtungen ist zufällig zu gestalten, wobei alle diese Richtungen gleichrangig sind. Als erstes sind deshalb alle freien Richtungen zu ermitteln und zu registrieren (Zeilen 340 bis 390). In Zeile 400 erfolgt daraus eine zufällige Auswahl, während die Zeilen 410 bis 430 die so bestimmte Richtung konkretisieren. Letztendlich kann die Barrikade in den Zeilen 440 und 450 errichtet werden. Endeabfrage und Endemeldung schließen das Hauptprogramm ab, selbstverständlich muß der Schluß eine Aussage über den Erfolg des Spielers bei der letzten Partie beinhalten. Für die Programmierung ist noch wichtig zu wissen, daß unbedingt verhindert werden muß, daß der Rechner nach dem Programmende noch einmal in das Unterprogramm gerät. Durch Überspringen des Unterprogramms, Zurückspringen an den Programmanfang oder, wie hier, durch Abbruch des Programms mit END ist das erreichbar.

```
10 REM BARRIKADE
20 WINDOW 0,31,0,39:CLS
30 PRINT :PRINT"    BARIKADE"
40 PRINT :PRINT"    VERSUCHE, SOVIELE KURSORBEWEGUNGEN"
50 PRINT"    WIE MOEGLICH AUSZUFUEHREN."
60 PRINT INK 23;AT(20,20);">ENTER<"
70 A$=INKEY$:IF A$="" THEN 70
80 CLS:N=0
90 PRINT CHR$(17);STRING$(39,"■")
100 FOR I=0 TO 31
110 PRINT AT(I,39);"■■"
120 NEXT I
130 PRINT AT(31,1);STRING$(39,"■");CHR$(18)
140 SP(1)=0:SP(2)=0:SP(3)=1:SP(4)=-1
150 ZP(1)=1:ZP(2)=-1:ZP(3)=0:ZP(4)=0
160 I=INT(RND(1)*30)+1
170 J=INT(RND(1)*38)+1
180 PRINT AT(I,J);"O"
190 REM MANUELLER ZUG
200 A$=INKEY$:IF A$="" THEN 200
210 IF A$<CHR$(8) THEN 200
220 IF A$>CHR$(11) THEN 200
230 Z=I:S=J
240 IF A$=CHR$(8) THEN S=S-1
250 IF A$=CHR$(9) THEN S=S+1
260 IF A$=CHR$(10) THEN Z=Z+1
270 IF A$=CHR$(11) THEN Z=Z-1
280 GOSUB 500
290 B=VPEEK(A)
300 IF B>0 THEN 200
310 PRINT AT(I,J);" "
320 N=N+1:I=Z:J=S
330 PRINT AT(Z,S);"O"
340 REM RECHNERZUG
350 F=0
360 FOR K=1 TO 4
370 Z=I+ZP(K):S=J+SP(K):GOSUB 500
380 B=VPEEK(A):IF B=0 THEN F=F+1
390 C(K)=F:NEXT K
400 F=INT(F*RND(1))+1
```

```
410 FOR K=1 TO 4
420 IF C(5-K)=F THEN L=5-K
430 NEXT K
440 Z=I+ZP(L):S=J+SP(L)
450 PRINT AT(Z,S);"■"
460 IF C(4)>1 THEN 200
470 PRINT"■GEFANGEN NACH";N;"SCHRITTEN"
480 END
490 REM PIXEL-RAM-ADRESSE
500 A=Z*256+S
510 IF Z/2>INT(Z/2) THEN A=A-192
520 IF S<32 THEN 550
530 A=Z*4+INT(Z/8)*480+S+8160
540 IF Z/2>INT(Z/2) THEN A=A+60
550 RETURN
```

3. Modifikationen

1. Das strenge Äußere dieses Programms läßt sich durch die Wahl von anderen Zeichen zur Darstellung der Spielfigur und der Blockade etwas mildern. Da aber viele Zeichen aus dem Grundvorrat in der obersten Bildzeile kein Bit belegen, muß dann die Belegungsabfrage auf eine tiefere Bildschirmzeile gerichtet werden, z.B. durch Erhöhung von A in Zeile 550, also etwa

550 A=A+32:RETURN

2. Etwas aufwendiger, dafür aber auch wesentlich reizvoller ist die Erzeugung von Zeichen nach eigenen Vorstellungen. Grundlage dazu bildet auch dafür das bereits vorhandene Unterprogramm zur Arbeit mit dem PIXEL-RAM.
Zunächst wird im 8×8-Raster ein Zeichen entworfen und dieses Zeile für Zeile, als Dualzahl aufgefaßt, als numerischer Wert dargestellt (Bild 2).
Mittels VPOKE läßt sich nun dieses Zeichen an der betreffenden Stelle direkt in den PIXEL-RAM eintragen, da ja in Zeile 380 die zugehörige Adresse bereits ermittelt wurde. Das

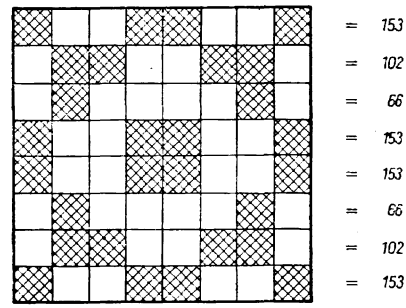


Bild 2

Programm ist also zu ergänzen bzw. zu ändern (s. Bild 3).
Bei der Wahl von anderen Zeichen sind gegebenenfalls die Hinweise zur 1. Modifikation zu berücksichtigen.

Und nun viel Vergnügen mit dem fertigen Programm – oder läßt sich vielleicht doch noch etwas daran verbessern?

Regelmäßige Vielecke

Das Hauptbetätigungsfeld der Computer ist die Verarbeitung von numerischen Informationen, also von Zahlen. Das Ergebnis der Bearbeitung sind

Bild 3

```
375 E(K)=A
420 IF C(5-K)=F THEN A=E(5-K)
440 VPOKE A,153:VPOKE A+128,102:VPOKE A+256,66:VPOKE A+384,153
450 VPOKE A+32,153:VPOKE A+160,66:VPOKE A+288,102:VPOKE A+416,153
```

zwangsläufig wiederum Zahlen, welche in unterschiedlicher Anordnung ausgegeben werden. Die Interpretation der Ergebnisse wird dabei in der Mehrzahl aller Fälle dem Nutzer selbst überlassen. Bei Computern mit grafischer Ausgabe bietet sich die Möglichkeit an, Resultate mit grafischen Mitteln anschaulicher darzustellen.
Zur Demonstration der grafischen Fähigkeiten soll ein regelmäßiges Vieleck auf dem Bildschirm dargestellt werden, in welches eine ineinandergeschachtelte Folge von gleichen Vielecken einzubeschreiben ist (Bild 4).

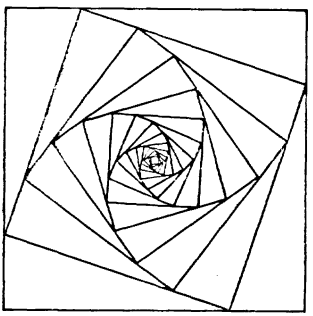


Bild 4. Beispiel Quadrat – Teilungsverhältnis 1:3

Bevor eine solche Aufgabe allgemeingültig gelöst werden kann, sind viele, zum Teil recht komplizierte Überlegungen anzustellen. Es ist deshalb erforderlich, mit der Lösung einer einfachen Teilaufgabe zu beginnen und diese stufenweise zu vervollkommen.

1. Zeichnen einer Strecke (Bild 5)

Bild 5

Der Rechner KC 85/2 arbeitet mit einem recht feinen Punktraster, welches näherungsweises Zeichnen von Strecken ermöglicht. Der Koordinatenraum umfaßt den Bereich $x = 0 \ldots 319$ und $y = 0 \ldots 255$. Alle Zeichnungen, die realisiert werden sollen, müssen in diesen Bereich eingeordnet bzw. transformiert werden.
Für jeden Punkt $P(x, y)$ der Strecke $\overline{P_1P_2}$ gilt:

$$\frac{y - y_1}{x - x_1} = \frac{y_2 - y_1}{x_2 - x_1}.$$

Die Arbeit im Punktraster hat nun die Besonderheit, daß in Abhängigkeit von der Steilheit der Strecke wahlweise die x-Achse oder die y-Achse verfolgt werden müssen, um zu einer geschlossenen Linie zu gelangen. Bei flachen Strecken (bis 45 Grad) verwendet man folglich

$$y = \frac{y_2 - y_1}{x_2 - x_1}(x - x_1) + y_1,$$

wobei x alle Werte zwischen x_1 und x_2 annimmt, während bei steileren Strecken die dazugehörige Umkehrfunktion zu verwenden ist:

$$x = \frac{x_2 - x_1}{y_2 - y_1}(y - y_1) + x_1.$$

Andernfalls wird die Darstellung entweder lückenhaft, bei Verfolgung der falschen Achse des Koordinatensystems, oder mit erhöhtem Rechenzeitaufwand belastet (z.B. bei Benutzung der Parameterdarstellung der Trägergeraden).
Zur einfacheren Handhabung dieses Programms im weiteren Entwicklungsverlauf wird das Zeichnen der Strecke als Unterprogramm realisiert:

```
30 CLS
300 X1=30:Y1=50:X2=300:Y2=100
310 GOSUB 410
390 END
```

Die Zeilen 400 bis 540 sind dazu aus der Gesamtdarstellung zu entnehmen.
Durch Variation der Zahlen in Zeile 300 lassen sich nun alle möglichen Strecken zeichnen und damit auch alle Zweige

des vorliegenden Programms austesten. Man muß hierbei natürlich auch senkrechte und waagerechte Strecken berücksichtigen. Selbstverständlich müssen die beiden Endpunkte der Strecke stets voneinander verschieden sein und innerhalb der Bildschirmgrenzen liegen. Ein diesbezüglicher Test ist in dem Unterprogramm weder enthalten noch vorgesehen – für die Einhaltung dieser Bedingungen ist der Programmierer, der das Unterprogramm nutzt, selbst verantwortlich.

2. Zeichnen eines Quadrates

Um ein Quadrat zu zeichnen, braucht man eigentlich nur vier Strecken gemäß obigem Vorbild zu erzeugen. Vorteilhafter ist es allerdings, hierfür einen Zyklus zu verwenden, insbesondere, da dieser leichter erweitert und verallgemeinert werden kann:

```
20 DIM X(4),Y(4)
40 N=4
50 X(1)=30:Y(1)=50:X(2)=130:Y(2)=50
60 X(3)=130:Y(3)=150:X(4)=30:Y(4)=150
```

Die Zeilen 270 bis 300 und 320 sind wiederum aus der Gesamtdarstellung zu entnehmen.
Alle diese Zeilen dienen der Ergänzung des bisher erarbeiteten Programms und ergeben bei gewissenhafter Ausführung ein Quadrat.

3. Zeichnen eines n-Ecks

Bei aufmerksamer Betrachtung des bisherigen bereitet eine Erweiterung auf beliebige geschlossene Streckenzüge sicherlich keine Schwierigkeiten mehr. Zeile 20 muß nur auf die Anzahl der erforderlichen Punkte erweitert werden, z. B. auf

20 DIM X (25), Y (25).

Danach sind nach obigem Vorbild N, X und Y entsprechend den Wünschen vorzugeben. Der Leser möge dies einmal selbst ausprobieren. Schwieriger wird es schon, wenn es um ein regelmäßiges n-Eck geht, wobei das Problem in erster Linie in der geometrischen Behandlung der Aufgabe zu suchen ist.

3.1. Geometrische Betrachtung

Die Zahl der Ecken ist n, also eine Variable. Das n-Eck soll so auf dem Bildschirm dargestellt werden, daß es eine vorgegebene Höhe ausfüllt. Es ist also zunächst die Kantenlänge in Abhängigkeit der Höhe darzustellen (Bild 6).

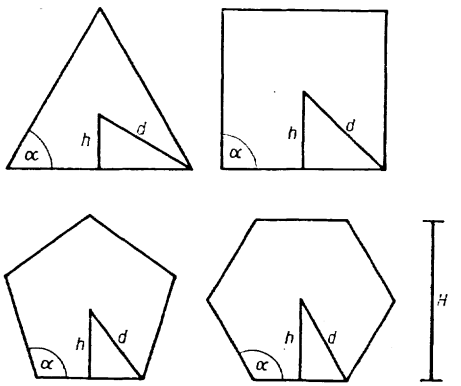


Bild 6

Es ist leicht zu beweisen, daß für den Innenwinkel die Gleichung

$$\alpha = \frac{(n-2)\,\pi}{n}$$

gilt. Da alle Winkelfunktionen im Rechner im Bogenmaß arbeiten, wird dieses hier ebenfalls ausschließlich zur Darstellung benutzt. (π entspricht 180 Grad). Weiterhin sind

$$h = \frac{a}{2} \tan \frac{\alpha}{2} \quad \text{und}$$

$$d = \frac{a}{2} : \cos \frac{\alpha}{2}.$$

Für gerades n gilt

$$H = 2h$$

$$H = a \tan \frac{\alpha}{2} \quad \text{bzw.}$$

$$a = \frac{H}{\tan \frac{\alpha}{2}}.$$

Für ungerades n ist analog

$$H = h + d$$

$$H = \frac{a}{2}\left(\tan \frac{\alpha}{2} + \frac{1}{\cos \frac{\alpha}{2}}\right)$$

oder

$$H = \frac{a}{2}\left(\frac{1 + \sin \frac{\alpha}{2}}{\cos \frac{\alpha}{2}}\right)$$

bzw.

$$a = 2H \frac{\cos \frac{\alpha}{2}}{1 + \sin \frac{\alpha}{2}}.$$

Es sei $H = 235$, also oben und unten etwa 10 Bildschirmzeilen als Rand. Dann liegt der Umkreismittelpunkt auf den Koordinaten

$$x_M = 160, \quad y_M = 10 + h.$$

Der Punkt P_1 sei unten links, für ihn gilt damit

$$x_1 = x_M - \frac{a}{2}, \quad y_1 = 10,$$

Alle übrigen Eckpunkte ergeben sich nun durch Drehung des ersten Eckpunktes um den Umkreismittelpunkt um jeweils einen Winkel von $2\,\pi/n$. Für eine Drehung um den Koordinatenursprung gelten die Formeln

$$x' = x \cos \varphi - y \sin \varphi$$

$$y' = x \sin \varphi + y \cos \varphi.$$

Daraus ist für einen anderen Drehpunkt abzuleiten:

$$x' = (x - x_M) \cos \varphi - (y - y_M) \sin \varphi$$

$$y' = (x - x_M) \sin \varphi + (y - y_M) \cos \varphi.$$

Mit $\varphi = 2\,\pi/n$ lassen sich mit diesen Formeln P_2 aus P_1, P_3 aus P_2 usw. bis P_n errechnen.

3.2. Rechentechnische Lösung

Im vorliegenden Programm wird wie folgt vorgegangen. Zuerst wird der Winkel α berechnet und davon die Seitenlänge a abgeleitet (Zeile 110 bis 140). Zur Vereinfachung der Rechnung bzw. zur Rechenzeiteinsparung erfolgen die Verschiebung des n-Ecks aus dem Koordinatenursprung in die Bildschirmmitte erst nach der Erzeugung der jeweiligen Ecke mittels des aufgeführten Rotationsverfahrens (Zeile 200), hingegen die Berechnung der Winkelfunktionen einmalig außerhalb des Zyklus (Zeile 180). Unter Beachtung dieser Hinweise sind die Zeilen 50 und 110 bis 240 in das bisher erarbeitete Programm zu übernehmen. Da die vorherige Belegung der Eckpunkte überflüssig geworden ist, werden die betreffenden Zeilen gelöscht (40, 60 und eigene Hinzufügungen des Lesers). Die Qualität der Darstellung des regelmäßigen Vielecks ist nun nur noch vom technischen Zustand des verwendeten Fernsehgerätes abhängig.

4. Verschachtelung

Hübsche Bilder entstehen mit dem Computer durch Variation und Überblendung eines Grundthemas. Solche Variationen lassen sich relativ einfach erzeugen, die Mannigfaltigkeit erbringt ein zyklischer Aufruf.

Im vorliegenden Beispiel soll jeweils ein gleichartiges Vieleck in das vorhandene einbeschrieben werden. Dazu ist eine mögliche Bildungsvorschrift, daß alle Seiten durch die neuen Eckpunkte im gleichen Verhältnis zu teilen sind. Am einfachsten geht dies mit Hilfe der Vektorrechnung zu beschreiben.

$$\overrightarrow{P_1'} = \overrightarrow{P_1} + \mu\,(\overrightarrow{P_2} - \overrightarrow{P_1}).$$

```
10 REM REGELMAESZIGE VIELECKE
20 DIM X(25),Y(25)
30 CLS
40 PRINT"AUSGABE VERSCHACHTELTER VIELECKE":PRINT
50 INPUT"  ECKENANZAHL: ";N
60 IF N<3 OR N>25 THEN 50
70 IF N>INT(N) THEN 50
80 PRINT
90 INPUT"  TEILUNGSVERHAELTNIS 1:";Q
100 IF Q<=0 THEN 90
110 AL=(N-2)*PI/N/2
120 IF INT(N/2)<N/2 THEN 140
130 A=235/TAN(AL):GOTO 150
140 A=470*COS(AL)/(1+SIN(AL))
150 ZX=-A/2:ZY=ZX*TAN(AL)
160 YM=10-ZY
170 FI=2*PI/N
180 SF=SIN(FI):CF=COS(FI)
190 FOR I=1 TO N
200 X(I)=ZX+160:Y(I)=ZY+YM
210 ZW=ZX
220 ZX=ZX*CF-ZY*SF
230 ZY=ZW*SF+ZY*CF
240 NEXT I
250 CLS:REM BEGINN DER ZEICHNUNG
260 FOR I=1 TO 15
270 X(0)=X(N):Y(0)=Y(N)
280 FOR J=1 TO N
290 K=J-1
300 X1=X(K):X2=X(J):Y1=Y(K):Y2=Y(J)
310 GOSUB 410
320 NEXT J
330 REM SCHRUMPFUNG
340 FOR J=N TO 1 STEP -1
350 X(J)=(Q*X(J)+X(J-1))/(Q+1)
360 Y(J)=(Q*Y(J)+Y(J-1))/(Q+1)
370 NEXT J,I
380 W$=INKEY$:IF W$="" THEN 380
390 GOTO 40
400 REM UP STRECKE ZEICHNEN
410 DX=X2-X1:DY=Y2-Y1
420 IF DX*DX>DY*DY THEN 490
430 SY=SGN(DY):A=DX/DY:B=X1-A*Y1
440 FOR T=Y1 TO Y2 STEP SY
450 S=A*T+B
460 PSET S,T,7
470 NEXT T
480 RETURN
490 SX=SGN(DX):A=DY/DX:B=Y1-A*X1
500 FOR S=X1 TO X2 STEP SX
510 T=A*S+B
520 PSET S,T,7
530 NEXT S
540 RETURN
```

Bild 7

Das läßt sich auch darstellen in der Form

$$\overrightarrow{P_1'} = \frac{\overrightarrow{P_1} + q\overrightarrow{P_2}}{1+q}.$$

Für eine innere Teilung muß μ einen Wert zwischen 0 und 1 annehmen. Im zweiten Fall spricht man auch von einem gewichteten Mittel, wobei q einen beliebigen Wert größer Null annehmen darf. Dieser zweite Weg wurde im vorliegenden Programm beschritten. Es ergeben

$q > 1$ Rechtsdrall,
$q = 1$ fortlaufende Halbierungen und
$q < 1$ Linksdrall in der Verschachtelung.

Durch Verwendung von q als Programmvariable ist es später möglich, zu verschiedenartigen Abbildungen auf dem Bildschirm zu kommen.
Das begonnene Programm ist nun zu ergänzen durch die Programmzeilen 90, 260, 340 bis 370 und 390, wobei letztere eine bisher verwendete END-Anweisung überschreibt.
Hiermit ist das Programm voll funktionsfähig, also im wesentlichen fertig-

gestellt. Es lohnt sich aber immer, das Programm erst noch zu komplettieren, bevor es endgültig abgelegt werden kann.

5. Kosmetik

Dieser Begriff hat sich eingebürgert für die Behebung von unwesentlichen Mängeln, die aber einer Fremdnutzung der Programme erhebliche Schwierigkeiten entgegenstellen können, also der Stolperstellen. Hierzu gehören eine einwandfreie Bedienerführung, vollständiger Test der Eingabewerte entsprechend dem zugelassenen Wertebereich, textliche Erläuterung der Rechenergebnisse und nicht zuletzt eine ausreichende Kommentierung des Programms zur Unterstützung nachfolgender Überarbeitungen, Ergänzungen bzw. zum leichteren Auffinden der Ursachen später entdeckter Fehler.
Bild 7 zeigt das so vervollständigte Programm. Durch Vergleich sind die betreffenden Stellen schnell auszumachen. Insbesondere ermöglicht Zeile 380, das fertige Bild ohne Einblendung einer Programmendemeldung zu betrachten. Durch Drücken einer beliebigen Taste geht es dann wieder an den Programmanfang zurück.

Literatur

[1] Jahn, F.: Alte deutsche Spiele. – Dresden, 1923
[2] Schubert, H.; Fitting, F.: Mathematische Mußestunden. – Berlin, 1943
[3] Kührt, H.: Alte und neue Brettspiele. – Weimar, 1951
[4] Rüger, B.: Rätsel, Jux und Zauberei. – Leipzig, 1958
[5] Steinhaus, H.: Kaleidoskop der Mathematik. – Berlin, 1959
[6] Domorâd, A. P.: Matematičeskie igry i razvlečeniâ. – Moskau, 1961
[7] Perelman, J. I.: Heitere Mathematik. – Berlin, 1962
[8] Rüger, B.: Du bist dran. – Leipzig 1962
[9] Lehmann, J.: Kurzweil durch Mathe. – Leipzig, 1981
[10] Perelman, J. I.: Unterhaltsame Aufgaben und Versuche. – Moskau, 1977
[11] Hirte, W.: Unsere Spiele. – Leipzig, 1982
[12] Bolchowitinow, W. N.; Koltowoi, B. I.; Lagowski, I. K.: Spaß für freie Stunden. – Leipzig, 1984
[13] Das Fünfzehnerspiel. – In: alpha 3 (20), 1986

Autor:
Dipl.-Math. Walter Görgens
Technische Universität
„Otto von Guericke"
Magdeburg, Sektion Informatik

Fortsetzung von Seite 23

Literatur

[1] Frank, M.; Lorenz, P.: Simulation diskreter Prozesse. – Leipzig: Fachbuchverlag, 1979
[2] Knocke, R.; Lohse, K.: SIMFOR – ein allgemeines Simulationssystem auf SKR und PC. – 15. Jahrestagung der WGMA. – Rostock, 1986
[3] Reitman, J.: Computer Simulation Application. – New York: John Wiley and Sons, 1971
[4] Henrikson, J. O.: The Development of GPSS/85. – Proc. 1985 Simulation Symposium. – In: Computer Soc. Press, 1985
[5] Preuss, F.: Forderungen an eine interaktive Nutzung von Simulationssystemen und ihre Realisierung durch Erweiterungen zum PS SIMDIS. – 1987. – Magdeburg, Techn. Hochsch., Diss.

Autoren:
Prof. Dr. sc. nat. Peter Lorenz
Dr. Thomas Schulze
Technische Universität
»Otto von Guericke« Magdeburg
Sektion Informatik

Fortsetzung von Seite 37

```
101C        D5          PUSH DE
101D        11 00 0F    LD DE, 0F 00    Zeitkonstante kurz
1020 M2:    1B          DEC DE          Abwärtszählen
1021        7A          LD A, D
1022        B3          OR E            D und E gleich Null
1023        20 FB       JRNZ M2         Weiter abwärtszählen
1025        D1          POP DE
1026        C9          RET
1027        D5          PUSH DE
1028        11 FF FF    LD DE, FFFFH    Zeitkonstante lang
102B M3:    1B          DEC DE          Abwärtszählen
102C        00          NOP             Auch ein NOP-Befehl
102D        00          NOP             braucht Zeit
102E        00          NOP
102F        7A          LD A, D
1030        B3          OR E            D und E gleich Null?
1031        20 F8       JRNZ M3         Weiter abwärtszählen
1033        D1          POP DE
1034        C9          RET
```

ISBN 3-343-00481-2

1. Auflage
Lizenznummer 114-210/3/89
LSV 1083
Verlagslektor: Helga Fago
Printed in GDR
Satz und Druck:
Messedruck Leipzig, Bereich Borsdorf
III-18-328
Redaktionsschluß: 15. 12. 1988
Bestellnummer: 547 509 3
00780

Kleinstrechner-TIPS/Hrsg. von Hans Kreul u. a. – Leipzig: Fachbuchverl., Heft 11. – 1. Aufl. – 1989. – 64 S.: 43 Bild.

Herausgeber:
Prof. Dr.-Ing. Hans Kreul, Technische Hochschule Zittau, Abt. EDV und Rechentechnik, Theodor-Körner-Allee 16, Zittau, 8800;
Prof. Dr. sc. techn. Thomas Horn,
Doz. Dr.-Ing. Wilhelm Leupold,
Informatik-Zentrum des Hochschulwesens an der Technischen Universität Dresden, Mommsenstr. 13, Dresden, DDR – 8027

Anschrift des Verlages:
VEB Fachbuchverlag Leipzig
PSF 67
Leipzig, DDR – 7031

Vorschau auf die nächsten Hefte

Schönfelder: Möglichkeiten der Programmierung statischer und dynamischer Pseudographik

Hamm: LDIR – ein Weg zur Einstimmung auf die Programmierung in Maschinensprache

Oelschlägel/Schulz: Anwendung des Kleinrechners zur Bestimmung des größten gemeinsamen Teilers und zur Bestimmung von Lösungen im Komplexen

Bogatz: Vollgraphik-Bildschirmansteuerung

Kühnel: Benchmarktests – eine Vergleichsmöglichkeit von Computerhard- und -software

Hamm: Grafik mit dem KC 85/1

Filla: Pixelgrafik – mehr Komfort für den KC 85/2

Schönfelder: Einsatz graphischer Bildschirmsysteme

Finck/Berndt: Nutzung von Kleincomputern in der Psychologieausbildung

Schönfelder: Drucken von pseudographischen Bildern

Pöschel: Römische Zahlen mit dem KC 85/1

Die Broschürenreihe

KLEINSTRECHNER-TIPS

behandelt

- Tendenzen und Theorien
- Informationen und Ideen
- Programme und Projekte
- Spaß und Spiel

und stellt sich das Ziel

- den Nutzer der Mikrorechentechnik aus allen Bereichen der Volkswirtschaft und dem Bildungswesen bei der Einarbeitung in die Informatik und Computertechnik zu unterstützen
- Entwicklungstendenzen der Informatik und Computertechnik vorzustellen und zur Erweiterung des Grundwissens beizutragen
- Anregungen für den Computereinsatz zu geben und Beispielprogramme für Kleincomputer zu veröffentlichen

um somit einem großen Kreis von Freunden der Informatik und Computertechnik zu helfen, sich moderner Hilfsmittel und Methoden zu bedienen.

ISBN 3-343-00481-2