7/93
64'er Markt&Technik
Juli 1993
öS 66,-/ sfr 7,80 hfl 9,25/ Lit. 10000 DM 7,80

Die Nummer 1 für C64 und C128

64'er

DAS

GRATIS-DISKETTE IM HEFT

! Exklusiv für alle 64'er-Leser

vollgepackt mit pädagogisch wertvollen **Lernspielen**

Grafik

Virtuelle Illusionen

■ Faszination Computergrafik

Test

Low-cost-Software

■ Da sprudelt die Software-Quelle!

Programm des Monats

GoDot: Profi-Bildmanipulator

■ konvertiert Amiga- und PC-Bilder
■ ermöglicht prima Bildeffekte
■ ist universell erweiterbar

Bauanleitung

Mini-Verstärker mit Maxi Leistung

• So wird der C 64 zum Sound-Künstler

Viele Tips & Tricks

Software-Corner:
• Heiße Kniffe zu Vizawrite
Basic-Corner:
• Sequentielle Dateien
Assembler-Corner:
• Perfektes IRQ-Handling
Profi-Corner:
• DYSP-Effekt
GeosWorkshop:
• Diagramme mit GeoChart

Grundlagen Vektorgrafik-Programmierung

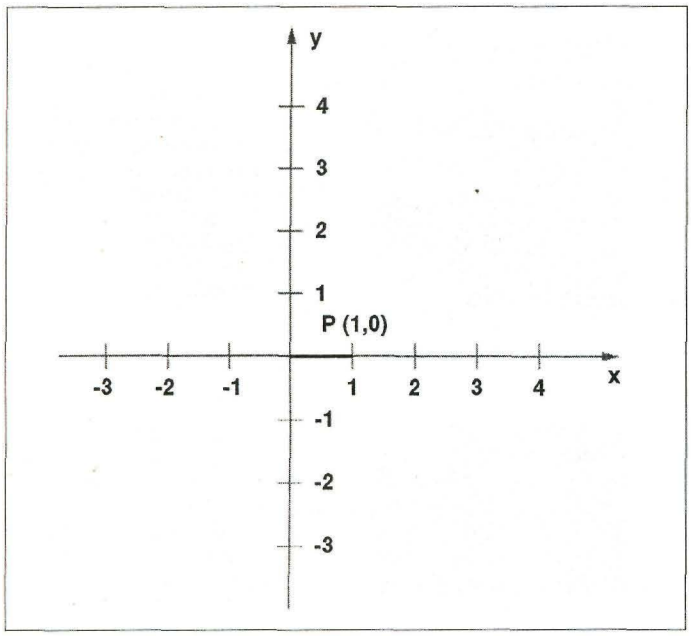
Vektorgrafik – bei diesem Begriff denkt man vielleicht zuerst an Stichwörter wie Großrechner, schnelle Prozessoren und viel Rechenpower. Daß Vektorgrafik auch auf dem C 64 möglich ist, beweist uns das Kultspiel »Elite«. Dieser Artikel soll uns in die Geheimnisse dieses phantastischen Gebiets der Computeranwendung einführen.

von Frank Schneider

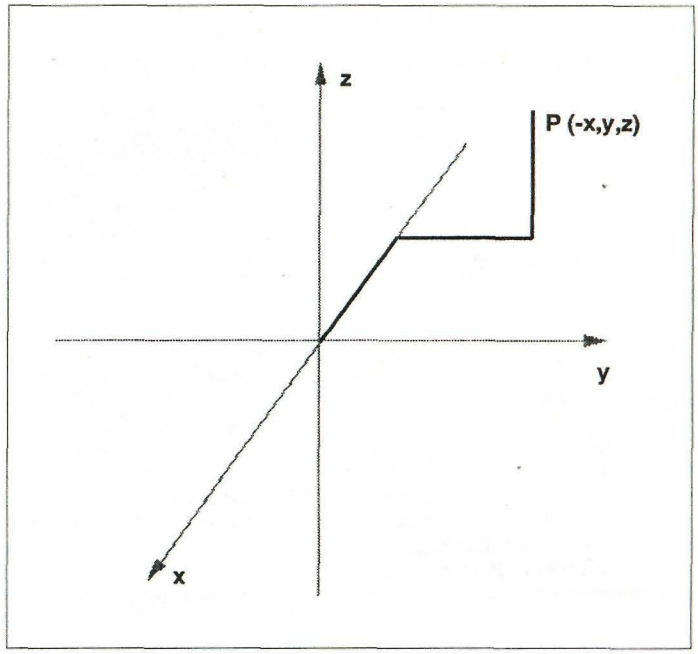
Vor den Erfolg haben die Götter den Schweiß gesetzt. Und so kommen wir auch nicht drum herum, uns zuerst mit einigen mathematischen Dingen und Hintergründen zu beschäftigen. Ferner sollte schon ein gewisses Maß am Verständnis für die Assemblersprache auf dem C 64 vorhanden sein, denn nur damit lassen sich die programmtechnischen und vor allem zeitkritischen Probleme sinnvoll lösen. Dieser Artikel soll, neben den Grundlagen für fortgeschrittene Programmierer, auch Ideen und Anregungen für eigene Programmierung von Vektorgrafik vermitteln und somit nicht als »abgeschlossenes Projekt« betrachtet werden.

Mathematische Grundlagen

Zunächst beschäftigen wir uns in diesem Artikel damit, wie man dreidimensionale Objekte, z.B. einen Würfel, auf dem Grafikbildschirm darstellen kann. Dazu ist es sinnvoll, das darzustellende

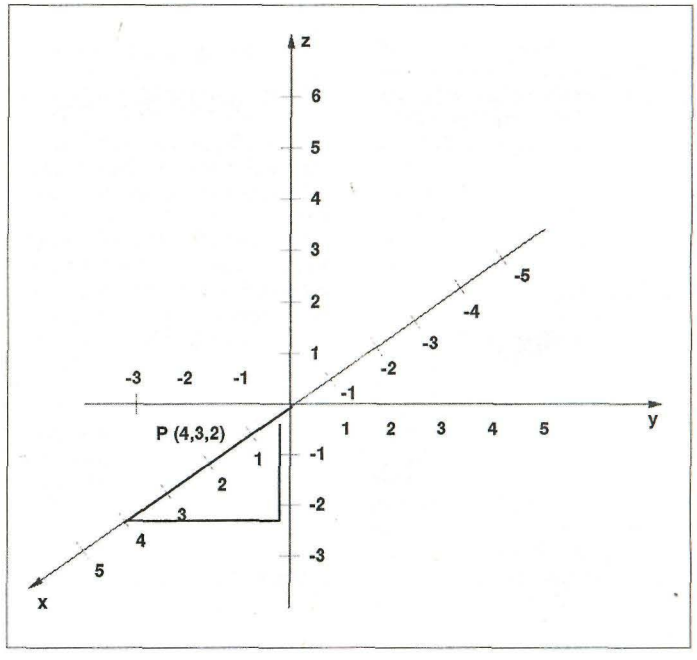


Ein Punkt im zweidimensionalen Koordinatensystem



Begibt man sich in den Raum kommt eine Achse dazu

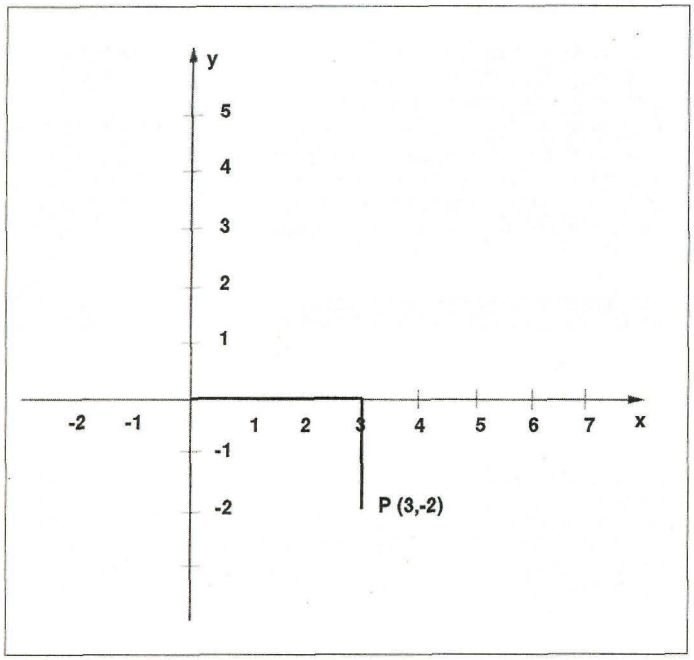
Objekt in einzelne Linien aufzugliedern, die dann nacheinander gezeichnet werden müssen. Eine zweite Alternative wäre, daß man das Objekt in einzelne Flächen oder sogenannte geschlossene Polygonzüge aufteilt. Damit könnte dann eine Logik zur plastischeren Darstellung von den Objekten programmiert werden, bei der verdeckte Linien nicht mehr gezeichnet werden. Mit dieser rechenintensiven und zu komplizierten Methode wollen wir uns hier allerdings nicht beschäftigen. Uns stellt sich jetzt also die Frage, wie wir Linien, die irgendwie im dreidimensionalen Raum liegen, auf den zweidimensionalen Grafikbildschirm zeichnen können. Hierzu ein Ausflug in die Mathematik.



Der Punkt im Raum wird durch drei Koordinaten bestimmt

Ein Punkt, der sich in einem zweidimensionalen Koordinatensystem befindet, ist eindeutig durch seine X- und seine Y-Koordinate bestimmt. Um also einen beliebigen Punkt P mit den beiden Koordinaten X und Y (Schreibweise P=(x,y)) in das Koordinatensystem einzuzeichnen, gehen wir zuerst x Einheiten auf der X-Achse und dann y Einheiten parallel zur Y-Achse entlang, gemäß den Vorzeichen von x und y. An dieser Stelle zeichnen wir den

Punkt P (Bild 1, das Ende der fetten Linie enspricht dem gewählten Punkt). Eine Linie bekommen wir, indem wir einen zweiten Punkt, nennen wir ihn Q, auf die gleiche Weise einzeichnen, wie zuvor für den Punkt P beschrieben und die beiden Punkte mit einem Lineal verbinden. Bis jetzt haben wir uns nur im zweidimensionalen Raum bewegt. Um das Ganze auf den dreidimensionalen Raum zu übertragen, benötigen wir eine dritte Koordinatenachse (s. Bild 2). Die Koordinatenachsen werden üblicherweise wie in Bild 2 zu sehen benannt. Die dritte Koordinatenachse liegt als Diagonale zwischen den beiden anderen Koordinatenachsen. Um nun einen Punkt bzw. eine Linie in dieses Koordinatensystem einzuzeichnen, gehen wir genauso vor, wie zuvor für das zweidimensionale System beschrieben. Auf der diagonalen Koordinatenachse bewegen wir uns genauso in positive oder negative Richtung wie zuvor auch, nur mit dem Unterschied, daß wir uns hier »schief« bewegen (s. Bild 3)



Mit unser Rechenformel transformiert man den Punkt aus Bild 3 ins zweidimensionale System

Eine ganz wichtige Erkenntnis ist nun, daß die schiefe Bewegung einer gleichzeitigen Bewegung in x- und y-Richtung im zweidimensionalen System entspricht. Wir können also die dritte Achse weglassen und die Bewegung auf der gedachten dritten Achse durch eine Bewegung in x- und y-Richtung im zweidimensionalen System ersetzen. Um eine gute und einfache dreidimensionale Darstellung zu bekommen, wird die Unterteilung der Einheiten auf der dritten Achse wie in Bild 3 zu sehen vorgenommen. Mit diesen Ergebnissen können wir nun eine einfache Formel für die Umrechnung von 3D nach 2D ableiten:

Zeichne den dreidimensionalen Punkt P mit P=(x,y,z) im zweidimensionalen Koordinatensystem mit den Achsen x' und y', das heißt zeichne P'=(x',y') mit x'=-0.5*x+y und y'=-0.5*x+z. Beispiel: Zeichne P=(4,3,2), x=4, y=3, z=2. Dies ergibt x'=-0.5*4+3=1 und y'=-0.5*4+2=0. Also P'=(1,0) im zweidimensionalen Koordinatensystem (s. Bild 3 und 4). Probieren Sie es doch einmal an eigenen Beispielen – es ist wirklich nicht schwer!

Auf diese Weise können dann auch Linien umgerechnet werden, indem zuerst die beiden Endpunkte der Linie umgerechnet werden und dann zwischen den beiden Punkten im zweidimensionalen Koordinatensystem eine Verbindungslinie gezeichnet wird.

Zusammenfassend können wir nun festhalten, daß wir mit der obigen Formel Punkte und Linien im dreidimensionalen Koordinatensystem umrechnen und in ein zweidimensionales System auf dem Grafikbildschirm - einzeichnen können, und das war ja unser Ziel.

In der Praxis

Nach so viel Theorie, die sich leider nicht vermeiden läßt, schreiten wir nun zur Praxis. Das Listing 1, welches Sie mit dem MSE V2.1 abtippen können, enthält unter anderem die komplette eben beschriebene Logik zur Umrechnung von 3D nach 2D. Enthalten sind ferner Routinen zum Ein- und Ausschalten der Grafik plus einem sehr schnellen Linienalgorithmus – dem Bresenham Algorithmus – der wohl einigen Lesern ein bekannter Begriff aus vergangenen Ausgaben der 64'er sein dürfte. Er wurde schon extra für mögliche noch zu programmierende Erweiterungen integriert, z.B. einer Animationsroutine für bewegte 3D-Objekte, bei der es dann besonders auf einen schnellen Linienalgorithmus ankommt. Das Maschinenprogramm ist so konzipiert, daß es sowohl von Basic als auch von Assembler aus aufgerufen werden kann. Nachfolgend sind die Aufrufe von Basic aus beschrieben:

SYS 49152,HF,ZF: Grafik einschalten, löschen und Farben setzen, wobei HF die Hintergrundfarbe und ZF die Zeichenfarbe angibt.

SYS 49155: Grafik ausschalten.

SYS 49158,C,x1,y1,z1,x2,y2,z2: 3D-Linie auf Grafikbildschirm zeichnen oder löschen. C=0 löscht eine Linie und C=1 zeichnet sie. Die sechs folgenden Daten geben den Anfangs- und Endpunkt der Linie im 3D-Raum an (im Bereich von -60 bis +60). Sollte bei eingeschalteter Grafik ein Fehler in den Bereichsgrenzen auftreten, dann geben Sie blind den Befehl ein, um die Grafik auszuschalten.

Die Aufrufe in Assembler sehen wie folgt aus:
Grafik einschalten, löschen und Farben setzen mit:

```
LDA #Wert
JSR $C040
```

Die oberen vier Bits des Akkus geben die Zeichenfarbe und die unteren vier Bits die Hintergrundfarbe an.
Grafik ausschalten mit:

```
JSR $C079
```

3D-Linie auf Grafikbildschirm zeichnen oder löschen mit:

```
CLC
```

oder

```
SEC
LDA #$A0
JSR $C0C7
```

Die Befehle CLC oder SEC bedeuten Linie löschen oder Linie zeichnen. Im Akku wird das HI-Byte der Grafikbasisadresse übergeben. In den Zeropageadressen $A4-$A9 müssen die sechs vorzeichenbehafteten Byte-Zahlen des Anfangs- und des Endpunkts der Linie im 3D-Raum stehen. Vorsicht: der Bereich wird nicht getestet wie bei dem Basic-Aufruf! Die Zeropageadressen $A4-$A9 werden durch die Routine nicht verändert. Deshalb muß man selbst im Programm die Werte dorthin schaffen.

Mit diesem Maschinenprogramm haben Sie nun ein Werkzeug in der Hand, um von Basic oder von Assembler aus dreidimensionale Objekte auf dem Grafikbildschirm darzustellen. Listing 1 zeigt ein Basicprogramm, das einige Möglichkeiten des Grafikmoduls demonstriert. Tippen Sie das Basic-Programm mit dem Checksummer ab und speichern Sie es auf Diskette. Um das Demoprogramm zu starten, laden Sie zuerst das Maschinenprogramm (Listing 2) absolut mit:

```
LOAD"VEKTORGRAFIK/OBJ",8,1
```

geben NEW ein und laden dann Listing 1. Dann kann das Demo (Basic-Programm) mit < RUN> gestartet werden. Der Bildschirm wird auf Grafik geschaltet und ein Quader mit räumlichen Diagonalen gezeichnet.

Wer sich noch intensiver mit dem Maschinenprogramm oder dem Bresenham-Algorithmus beschäftigen möchte, der sei auf den dokumentierten Quelltext und die Infobox zum Quelltext hingewiesen. Informationen zu Vektoren oder zur Vektorrechnung finden Sie in der Ausgabe 10/92 der 64'er in der Anleitung zu meinem Programm »Analytische Geometrie v1.2'« oder in dem im Klett-Verlag erschienen Buch »Analytische Geometrie – Leistungskurs« von den Autoren Lambacher und Schweizer (ISBN-Nr. 3-12-739170-6). (lb)

Infobox zum Quelltext

Der Quelltext ist so konzipiert, daß der Grafikbildschirm ab $A000 und das Farb-RAM ab $8C00 liegen. Somit ist der Basic-Speicher für eigene Programme von $A000 auf $8C00 herabgesetzt, was aber bei der Benutzung dieses Moduls noch allemal reichen dürfte. Doch nun zu der zentralen Routine zum Zeichnen von Linien im 3D-Raum auf dem Grafikbildschirm.

Zeile 1260-1460: Schleife zum Einlesen der Parameter aus dem Basictext. Die Fließkommazahlen werden gerundet, in vorzeichenbehaftete Byte-Zahlen im Bereich -60 bis +60 ($C4-$3C) umgewandelt und in die Zeropageadressen $A4-$A9 abgespeichert.

Zeile 1540-1750: Schleife zum Umwandeln der beiden übergebenen Punkte in ein Delta x und ein Delta y für jeden der beiden Punkte, wobei die Deltas vom linken oberen Grafikpunkt aus gemessen sind – jeweils in Anzahl Grafikpunkte. Dabei wird die im Artikel angesprochene Formel zur Umrechnung von 3D nach 2D verwendet. Der Koordinatenursprung ist der Mittelpunkt des Grafikbildschirms (160,100). Die Deltas des ersten Punkts kommen nach $AA/$AB und die des zweiten Punkts nach $AD/$AE.

Zeile 1780-1930: Routine zum Berechnen von Delta x und Delta y zwischen den zuvor berechneten beiden Grafikpunkten des Anfangs- und Endpunktes der zu zeichnenden Linie. Die Deltas werden in $B1/$B2 und die Vorzeichen der Deltas in $B3/$B4 abgespeichert.

Zeile 1950-1990: Berechnet die Bit-Maske des ersten zu setzenden oder zu löschenden Grafikpunktes. Die Bit-Maske wird in der Zeropage-Adresse $AC abgelegt.

Zeile 2010-2160: Berechne Zeiger auf Byte des ersten zu setzenden oder zu löschenden Punktes. Der Zeiger wird in $AA/$AB abgespeichert.

Zeile 2180-2510: Berechne Daten für den Bresenham-Algorithmus. In $AD/$AE steht die konstante erste Richtung. Sie ergibt sich aus dem größten Wert von Delta x und Delta y. $AF/$B0: die variable zweite Richtung. Sie wird entsprechend dem Linienalgorithmus gesetzt oder bleibt auf 0. $B1 ist die Vergleichszahl für laufende Summe. Erreicht die laufende Summe diese Zahl, so wird die zweite Richtung gesetzt. $B2 ist der Summand für laufende Summe. Dieser Wert wird in jedem Schleifendurchgang zu der laufenden Summe in $B3 (Startwert ist 0, siehe bei $B1) addiert. In $B4 befindet sich der Abwärtszähler für noch zu setzende Anzahl von Grafikpunkten. Ist er 0, so ist die Linie fertig gezeichnet. In $B5 steht der Wert für die zweite Richtung. Er ist genau so wie der Wert für die konstante erste Richtung -1 oder +1 entweder für die x- oder die y-Koordinate.

Zeile 2540-2710: Berechne die laufende Summe, vergleiche mit der Vergleichszahl und setze gegebenenfalls die zweite Richtung.

Zeile 2730-3330: Berechnet Zeiger und Bit-Maske gemäß den Werten der ersten und der zweiten Richtung und beachte ferner den speziellen Aufbau der Bitmap (gleich dem Grafikbildschirm).

Zeile 3350-3500: Setze oder lösche den berechneten Grafikpunkt und beende die Zeichenroutine, wenn der Abwärtszähler 0 erreicht hat. Diese Routine kann von Assembler aufgerufen zirka 75 Linien pro Sekunde zeichnen. Sie ist damit eine schnelle Linienroutine für eine optionale Animationsroutine für bewegte Vektorgrafikobjekte.

Listing 1: Beispielprogramm zu Listing 1

```
100 PRINT"{CLR,2DOWN,3SPACE}DEMO ZUR VEKTO
    RGRAFIK"                                <186>
110 PRINT"{4SPACE}(W) BY F. SCHNEIDER{2DOW
    N}"                                     <152>
120 INPUT" ZAHL (VON -60 BIS +60)";K        <208>
130 K=INT(K+0.5)                            <116>
140 IF K<-60 OR K>60 THEN PRINT"{2UP}":GOT
    O 120                                   <003>
150 REM                                     <212>
160 SYS 49152,1,0                           <153>
170 FOR I=-K TO K STEP K/5                  <073>
180 SYS 49158,1,K,K,-K,-K,K,I               <025>
190 SYS 49158,1,-K,K,-K,-K,-K,I             <139>
200 SYS 49158,1,-K,-K,-K,K,-K,I             <255>
210 SYS 49158,1,K,-K,-K,K,K,I               <235>
220 NEXT I                                  <048>
230 SYS 49158,1,K,K,K,-K,K,K                <178>
240 SYS 49158,1,-K,K,K,-K,-K,K              <254>
250 SYS 49158,1,-K,-K,K,K,-K,K              <208>
260 SYS 49158,1,K,-K,K,K,K,K                <054>
270 POKE 198,0:WAIT 198,1:POKE 198,0        <119>
280 SYS 49155                               <213>
290 GOTO 120                                <252>
```

Listing 2: Die Routinen zum Zeichnen räumlicher Linien

```
"vektorgrafik/obj"          c000 c257

c000: ipwl 7scy xafh 6pgj 4kun uzwj dk
c00f: 4kun uzwj 4i3j rmte 7epj rm3e bn
c01e: 7elf ah77 rq43 qtg2 3251 utgv 7o
c02d: 17p7 axtj edgx k5y7 7cqh tbpj a4
c03c: ahed k54b 7bph yaw2 ps5j daa7 e4
c04b: dg7j d7e7 tbco wag3 thpj r7a7 dd
c05a: dg7j z7f4 eg57 r7lm 7cnz zffp c4
c069: edgp rndm cchj zdnp adph zdnp e3
c078: lbvq ctai 25fq ctei b5fq qtei 76
c087: r5fp awk7 iqdk dh77 3kp7 elgv 7y
c096: qie7 qamj dc5z 3hdj utpd sna7 ax
c0a5: ujxz mjtt trui qx77 57c5 qtfx dz
c0b4: 4d7t s65i gvxm aimj md73 ratp bn
c0c3: zvtz 7j7h i7pa kpeb 7bzz irl7 b6
c0d2: mhh7 dzh7 uad6 5fci tdlg kilu ch
c0e1: ujla pzkd gczz memk 4ctn qx7f a6
c0f0: rcnj d7ix vvv6 kjuh qiud s7lu de
c0ff: vlei qd7d ig6v r7du vgea ay4e gu
c10e: uhtp oju4 cwah kkee ult6 prrj 7q
c11d: ujrz tjox pvuj kjyi 73bz t6ie gg
c12c: xjbz tzc4 d2ah kj4b 7bp7 aimq gn
c13b: xvyi 7brh tvyd qimt tzy2 7bue d5
c14a: vidj kljh tvyz imdf uvbj 4amu a2
c159: mbb2 eyur mbb2 cyuq pvzj r7de fc
c168: vnbz 6amp ip71 ejh7 pvwx kl7x dd
c177: tvyv klup 7zb2 gqmq r7hn klle er
c186: vnr2 kium z7bh kk6p 7jb2 ahpb 7m
c195: vvv6 7iip bicj yd76 lzvj kjsi cy
c1a4: abbz ud7u 3zu2 7dhf urh7 ziul g6
c1b3: gbrz uzhh pvuk 77vf unzz 43az a6
c1c2: f7m5 mjvp 7ksj wimj edc4 7k7x cl
c1d1: tvuf rnde ujrz vzha pvuy 7gme dt
c1e0: ukh7 equk xzuj kjqi 74dp ot7m br
c1ef: gbrz uzix pvuj kj6i 7fbz wrvj fc
c1fe: bbjr qiml t77k 7aji 63xz tk7q fh
c20d: ujhz uimt 57cl mm7h iqxl bs72 ds
c21c: xb7d 7h7p a7b7 d7h7 7a77 c77b dr
c22b: x7ap 7aj7 7z77 op7h 77ed 7b37 ea
c23a: as77 z77o h7hh 7dn7 bh7a hp7u ej
c249: p7kl 7ex7 ce7a u772 x7n7 7gw6 ep
```

Listing 3: Für alle interessierten Programmierer die Plotroutine für räumliche Linien im Quelltext

```
100 ASSEMBLER:OPT OO:*= $C000
110 ;
120 ;
130 ;  3D-LINE-ROUTINE
140 ;
150 ;
160 VEKTRAM    = $A4    ;6 BYTES
170 FREEZERO   = $FB    ;4 BYTES
180 FARBRAM    = $8C00
190 PICTURE    = $A000
200 ;
210 ;
220 ;  SPRUNGTABELLE
230 ;
240 JMP GRAPHICSON
250 JMP GRAPHICSOFF
260 JMP LINE
270 ;
280 NOP:NOP:NOP
290 NOP:NOP:NOP
300 NOP:NOP:NOP
310 NOP:NOP:NOP
320 ;
330 ;
340 ;  SUBROUTINEN
350 ;
360 ROMOFF SEI
370 LDA #$36
380 STA $01
390 RTS
400 ;
410 ROMON LDA #$37
420 STA $01
430 CLI
440 RTS
450 ;
460 FILLMEM LDY #0
470 L000 STA (FREEZERO+0),Y
480 INY
490 BNE L000
500 INC FREEZERO+1
510 DEX
520 BNE L000
530 RTS
540 ;
550 ;
560 ;  MAINROUTINEN
570 ;
```


Dieses Symbol zeigt an, welche Programme auf Diskette erhältlich sind

Listing für den 24-Nadeltreiber zu Printmaster

```
580 GRAPHICSON JSR $E200
590 TXA                 ;HINTERGRFARBE
600 AND #%00001111
610 STA FREEZERO+0
620 JSR $E200
630 TXA                 ;ZEICHENFARBE
640 ASL:ASL
650 ASL:ASL
660 EOR FREEZERO+0
670 LDX #<FARBRAM
680 LDY #>FARBRAM
690 STX FREEZERO+0
700 STY FREEZERO+1
710 LDX #4              ;FARBRAM
720 JSR FILLMEM         ;INITIALISIEREN
730 LDX #<PICTURE
740 LDY #>PICTURE
750 STX FREEZERO+0
760 STY FREEZERO+1
770 LDX #32
780 LDA #0              ;HIRESPAGE
790 JSR FILLMEM         ;INITIALISIEREN
800 ;
810 LDA 56576           ;GRAFIK
820 AND #252            ;EINSCHALTEN
830 ORA #1
840 STA 56576
850 LDA 53272
860 AND #15
870 ORA #(16*3+8)
880 STA 53272
890 LDA 53265
900 ORA #32
910 STA 53265
920 RTS
930 ;
940 GRAPHICSOFF LDA 53265
950 AND #223
960 STA 53265
970 LDA #23
980 STA 53272
990 LDA #151
1000 STA 56576
1010 RTS
1020 ;
1030 ;
1040 ;  3D-LINE-ROUTINE
1050 ;
1060 ;    $A4-$A9 -> 2 3D-VEKTOREN
1070 ;    $AA-$B5 -> VERSCH. GENUTZT
1080 ;      U.A. BRESENHAM-ALGORITHMUS
1090 ;    $AA-$AB -> VEKTOR IN PICTURE
1100 ;         $AC -> BIT D. PUNKTES
1110 ;    $AD-$B0 -> PUNKTRICHTUNGEN
1120 ;         $B1 -> VERGLEICHSZAHL
1130 ;         $B2 -> SUMMAND
1140 ;         $B3 -> LAUFENDE SUMME
1150 ;         $B4 -> ABWAERTSZAEHLER
1160 ;         $B5 -> ZWEITE RICHTUNG
1170 ;
1180 ERROR JMP $B248
1190 ;
1200 LINE JSR $E200
1210 CPX #2
1220 BCS ERROR
1230 TXA
1240 LSR
1250 PHP                ;ZEICHNEN/LOESCHEN
1260 L001 STA $AA       ;SCHLEIFE ZUM
1270 JSR $AEFD          ;EINLESEN DER
1280 JSR $AD8A          ;6 KOEFFIZIENTEN
1290 JSR $B849
1300 JSR $B1AA
1310 LDX $AA
1320 STY VEKTRAM,X
1330 TAX
1340 TYA
1350 CPX #0
1360 BEQ L002
1370 INX
1380 BNE ERROR
1390 SBC #1
1400 EOR #$FF
1410 L002 CMP #61
1420 BCS ERROR
1430 LDA $AA
1440 ADC #1
1450 CMP #6
1460 BCC L001
1470 LDA #>PICTURE
1480 PLP
1490 ;
1500 LINE2 PHP          ;EINSPRUNG FUER
1510 PHA                ;3D-LINE, WENN
1520 JSR ROMOFF         ;KOEFFIZIENTEN
1530 LDX #0             ;SCHON IN ZERO-
1540 L003 LDA VEKTRAM,X;PAGE STEHEN
1550 CMP #$80
1560 ROR
1570 BPL L004           ;3D -> 2D
1580 ADC #0             ;BERECHNE DX UND
1590 L004 TAY           ;DY FUER ANFANGS
1600 EOR #$FF           ;UND ENDPUNKT
1610 CLC                ;AUS DEN VOR-
1620 ADC #161           ;HANDENEN 6 VOR-
1630 CLC                ;ZEICHENBEHAF-
1640 ADC VEKTRAM+1,X    ;TETEN KOEFFI-
1650 STA $AA,X          ;ZIENTEN DER
1660 TYA                ;BEIDEN VEKTOREN
1670 CLC
1680 ADC #100           ;ERGEBNIS NACH
1690 SEC                ;$AA/$AB UND
1700 SBC VEKTRAM+2,X    ;$AD/$AE
1710 STA $AB,X
1720 INX:INX
1730 INX
1740 CPX #6
1750 BCC L003
1760 ;
1770 LDX #1             ;BERECHNE DX, DY
1780 L005 SEC           ;UND DEREN VOR-
1790 LDA $AD,X          ;ZEICHEN
1800 SBC $AA,X          ;ZWISCHEN AN-
1810 TAY                ;FANGS- UND END-
1820 TXA                ;PUNKT DER LINIE
1830 ROR
1840 EOR #$81           ;ABS(DX),ABS(DY)
1850 STA $B3,X          ;-> $B1/$B2
1860 ASL                ;VOR(DX),VOR(DY)
1870 TYA                ;-> $B3/$B4
1880 BCC L006
1890 EOR #$FF
1900 ADC #0
1910 L006 STA $B1,X
1920 DEX
1930 BPL L005
1940 ;
1950 LDA $AA            ;BERECHNE BIT-
1960 AND #%00000111     ;MASKE FUER 1.
1970 TAX                ;ZU SETZENDEN/
1980 LDA BITS,X         ;ZU LOESCHENDEN
1990 STA $AC            ;GRAFIKPUNKT
2000 ;
2010 LDA $AB            ;BERECHNE ZEIGER
2020 AND #%11111000     ;AUF BYTE DER
2030 LSR                ;HIRESPAGE, IN
2040 LSR                ;DEM DER ERSTE
2050 TAX                ;ZU SETZENDE/
2060 LDA $AA            ;ZU LOESCHENDE
2070 AND #%11111000     ;GRAFIKPUNKT
2080 STA $AA            ;STEHT
2090 LDA $AB
2100 AND #%00000111     ;ZEIGER IN
2110 ORA $AA            ;$AA/$AB
2120 ADC MULTAB+0,X
2130 STA $AA
2140 PLA                ;HI-B GRAF.PAGE
2150 ADC MULTAB+1,X
2160 STA $AB
2170 ;
2180 LDX #0             ;HOLE DIE VER-
2190 LDY #0             ;GLEICHSZAHL,
2200 LDA $B1            ;DEN ABWAERTS-
2210 CMP $B2            ;ZAEHLER, DEN
2220 BCC L007           ;SUMMANDEN UND
2230 PHA                ;DAS VORZEICHEN
2240 LDA $B2            ;FUER DIE ZWEITE
2250 PHA                ;RICHTUNG IN AB-
2260 LDA $B4            ;HAENGIGKEIT VON
2270 LDX $B3            ;MAX(DX,DY) FUER
2280 ;                  ;BRESH.-ALGORIT.
2290 .
2300 BCS L008           ;UNBEDINGT
2310 L007 LDA $B2
2320 PHA
2330 LDA $B1
2340 PHA
2350 LDA $B3
2360 LDY $B4
2370 ;
2380 L008 STX $AD       ;SETZE D. KONST.
2390 STY $AE            ;1. RICHTUNG,
2400 STA $B5            ;VORZ. 2.RICHT.,
2410 PLA
2420 STA $B2            ;DEN SUMMANDEN,
2430 INC $B2
2440 PLA
2450 STA $B1            ;DIE VERGL.ZAHL,
2460 INC $B1
2470 STA $B4            ;DEN ABW.ZAEHL.,
2480 LDA #0
2490 STA $B3            ;DIE LAUF. SUMME
2500 STA $AF            ;UND DIE VARIAB.
2510 STA $B0            ;2. RICHTUNG
2520 JMP L020
2530 ;
2540 L009 LDA #0        ;BRESENHAM-AL-
2550 STA $AF            ;GORITHMUS
2560 STA $B0
2570 CLC
2580 LDA $B3
2590 ADC $B2
2600 BCS L010
2610 STA $B3
2620 CMP $B1
2630 BCC L012
2640 L010 SBC $B1
2650 STA $B3
2660 LDA $B5
2670 LDX $AD
2680 BNE L011
2690 STA $AF
2700 BEQ L012           ;UNBEDINGT
2710 L011 STA $B0
2720 ;
2730 L012 LDX #2
2740 L013 LDA $AD,X
2750 BEQ L015
2760 BMI L014
2770 LSR $AC            ;VERAENDERE
2780 BCC L015           ;ZEIGER UND BIT-
2790 ROR $AC            ;MASKE FUER
2800 LDA $AA            ;X=X+1
2810 ADC #8
2820 STA $AA
2830 BCC L015
2840 INC $AB
2850 BCS L015           ;UNBEDINGT
2860 ;
2870 L014 ASL $AC       ;VERAENDERE
2880 BCC L015           ;ZEIGER UND BIT-
2890 ROL $AC            ;MASKE FUER
2900 SEC                ;X=X-1
2910 LDA $AA
2920 SBC #8
2930 STA $AA
2940 BCS L015
2950 DEC $AB
2960 ;
2970 L015 LDA $AE,X
2980 BEQ L019
2990 BMI L017
3000 INC $AA            ;VERAENDERE
3010 BNE L016           ;ZEIGER UND BIT-
3020 INC $AB            ;MASKE FUER
3030 L016 LDA $AA       ;Y=Y+1
3040 AND #%00000111
3050 BNE L019
3060 CLC
3070 LDA $AA
3080 ADC #<312
3090 STA $AA
3100 LDA $AB
3110 ADC #>312
3120 STA $AB
3130 BCC L019           ;UNBEDINGT
3140 ;
3150 L017 LDA $AA       ;VERAENDERE
3160 BNE L018           ;ZEIGER UND BIT-
3170 DEC $AB            ;MASKE FUER
3180 L018 DEC $AA       ;Y=Y-1
3190 LDA $AA
3200 AND #%00000111
3210 EOR #%00000111
3220 BNE L019
3230 SEC
3240 LDA $AA
3250 SBC #<312
3260 STA $AA
3270 LDA $AB
3280 SBC #>312
3290 STA $AB
3300 ;
3310 L019 DEX           ;2 MAL SCHLEIFE
3320 DEX                ;DURCHLAUFEN
3330 BPL L013           ;FUER X UND Y
3340 ;
3350 L020 PLP
3360 LDA $AC
3370 LDY #0
3380 BCS L021
3390 EOR #$FF
3400 AND ($AA),Y        ;PUNKT LOESCHEN
3410 BYT $2C            ;ODER
3420 L021 ORA ($AA),Y   ;SETZEN
3430 STA ($AA),Y
3440 LDA $B4
3450 BEQ L022
3460 DEC $B4
3470 PHP
3480 JMP L009
3490 ;
3500 L022 JMP ROMON
3510 ;
3520 BITS BYT $80,$40,$20,$10
3530 BYT $08,$04,$02,$01
3540 ;
3550 MULTAB BYT 0,0,<320,>320
3560 BYT <0640,>0640,<0960,>0960
3570 BYT <1280,>1280,<1600,>1600
3580 BYT <1920,>1920,<2240,>2240
3590 BYT <2560,>2560,<2880,>2880
3600 BYT <3200,>3200,<3520,>3520
3610 BYT <3840,>3840,<4160,>4160
3620 BYT <4480,>4480,<4800,>4800
3630 BYT <5120,>5120,<5440,>5440
3640 BYT <5760,>5760,<6080,>6080
3650 BYT <6400,>6400,<6720,>6720
3660 BYT <7040,>7040,<7360,>7360
3670 BYT <7680,>7680
3680 ;
```


Dieses Symbol zeigt an, welche Programme auf Diskette erhältlich sind