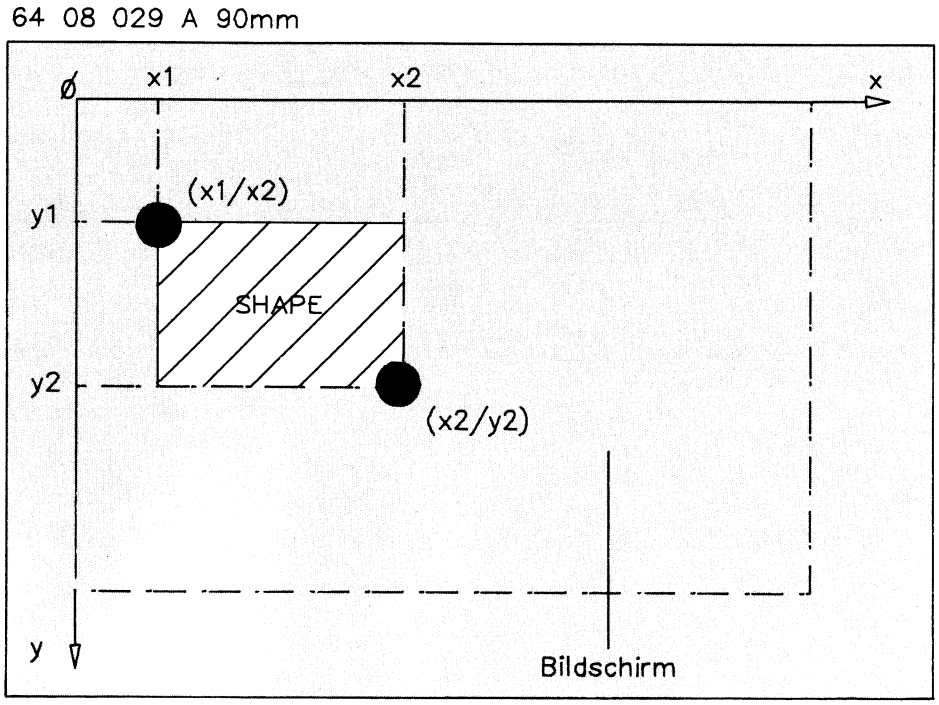

Bild 1.
Das schraffierte Rechteck kennzeichnet das spätere Shape

Shapes auf dem C16

Zwei Grafikbefehle sind es, die manchem C16-Benutzer einige Rätsel aufgeben. Beide haben mit den Shapes zu tun: SSHAPE und GSHAPE werden Ihnen hier vorgestellt.

Eine wichtige Kaufentscheidung bei Commodore-Computern sind ihre ausgezeichneten Grafikfähigkeiten. Doch leider ist ihnen teilweise recht schwer beizukommen. So etwa bei den Shapes. Hier erfahren Sie, wie Sie mit den Grafik-Befehlen SSHAPE und GSHAPE umgehen müssen. Doch zuerst: Was sind Shapes?

Wie so vieles aus der Computerkultur stammt auch dieses Wort aus dem Angelsächsischen: Shape heißt ins Deutsche übersetzt soviel wie Form, Gestalt oder Umriß. Hier bezeichnet Shape einen genau definierten Bildschirmausschnitt, der gespeichert und wiederverwendet werden kann. Wie das mit dem C16 geschieht, soll nun erklärt werden.

Die Shape-Befehle

1. SSHAPE

Dies ist der Befehl, mit dem ein Bildschirmbereich in einen String eingelesen werden kann. Seine Syntax ist

```
SSHAPE A$,X1,Y1,X2,Y2
```

A$ ist hier eine beliebige Stringvariable, auch ein Array kann Verwendung finden. X1 bis Y2 gibt die Eckkoordinaten des Rechteckes an, dessen Inhalt als Shape definiert werden soll. Bild 1 soll das etwas verdeutlichen.

(X1,Y1) gehören zum linken oberen und (X2,Y2) zum rechten unteren Eckpunkt. Es ist auch möglich, X2 und Y2 wegzulassen. In diesem Fall werden durch den SSHAPE-Befehl einfach die aktuellen Koordinaten des Grafik-Cursors eingesetzt.

Falls es noch unklar sein sollte: Im Gegensatz zu den Sprites anderer Commodore-Computer, die jedem Bildschirm-Modus trotzen (also sowohl als HiRes- oder Multicolorsprites im Text- als auch im Hochauflösungs- und Multicolormodus auftauchen können), sind Shapes fest mit dem aktuellen Modus verbunden. Ein Shape ist ein Grafikobjekt und kann daher auch nur in den Grafikmodi auftreten und definiert werden. Also in denen, die durch GRAPHIC erzeugt werden, wobei n von 1 bis 4 geht. Außerdem ist ein Multicolorshape auch nur im Multicolormodus ein solches. Schaltet man beispielsweise mittels GRAPHIC1 in den normalen Hochauflösungsmodus um, wird auch unser Shape nur als hochaufgelöstes sichtbar.

Wie wir uns aus all dem schon fast denken können, ist das Koordinatensystem, auf das sich die Angaben X1 bis Y2 beziehen, das im jeweiligen Modus gültige. Im Normalfall also im HiRes-Modus X von 0 bis 319 und Y von 0 bis 199. Im Multicolormodus geht X nur von 0 bis 159.

A$ ist ein String. Für Strings gilt beim C16 die Begrenzung auf maximal 255 Byte. Ein durch SSHAPE in einen String zu schreibendes Gebiet darf also eine gewisse Größe nicht überschreiten. Bezeichnen wir als

DX = X2 - X1 und als
DY = Y - Y1

dann gilt für die Bytemenge im String im Hochauflösungsmodus die Formel:

```
Lh=INT((ABS(DX)+1)/8+.99)*(ABS(DY)+1)+4
```

Im Multicolormodus gilt statt dessen die Gleichung:

```
Lm=INT((ABS(DX)+1)/4+.99)*(ABS(DY)+1)+4
```

Die Addition von 4 am Ende der Formeln ergibt sich durch 4 Byte, die den Schluß des Shape-String, bilden und Steuergrößen enthalten.

Sollte dann, trotz all dieser Berechnungsmöglichkeiten - man wendet ja nicht jedesmal die Formeln an - der zu speichernde Shape größer als 255 Byte werden, dann meldet unser Computer einen STRING TOO LONG ERROR.

Einen Aspekt des SSHAPE-Befehls müssen wir noch besprechen, weil es da einige Verwirrung geben kann: Wie reagiert SSHAPE auf ein durch SCALE verändertes Bild-

Fortsetzung auf Seite 46

Fortsetzung von Seite 44

schirmsystem? Die Koordinatenangabe bei SSHAPE muß ebenfalls in den aktuell gültigen Werten geschehen. Man muß sich also nicht um die Organisation des Koordinatensystems kümmern, denn durch SSHAPE wird automatisch die Umrechnung auf den normalen Bildschirm vorgenommen. Lediglich bei der Berechnung der Stringlänge müssen einige Korrekturen vorgenommen werden.

Nehmen wir an, wir hätten durch SCALE 1 ein System definiert, dann liegen (im Hochauflösungsmodus) die Vergrößerungsfaktoren

FX = 1023/320
FY = 1023/200

vor und die Beziehungen für die Stringlänge lauten dann:

Hochauflösungsmodus

```
Lh=INT((ABS(DX*320/1023)+1)/8+.99)*(ABS(DY*200/1023)+1)+4
```

Multicolormodus

```
Lm=INT((ABS(DX*160/1023)+1)/4+.99)*(ABS(DY*200/1023)+1)+4
```

DX und DY sind dabei im aktuellen Koordinatensystem geltende Werte.

Begnügen wir uns nun mit dem SSHAPE-Befehl (es gäbe noch einiges zu untersuchen) und wenden wir uns dem Gegenstück, nämlich dem GSHAPE-Befehl zu.

2. GSHAPE

Ein durch SSHAPE im String gespeichertes Bild kann durch diesen Befehl nun wieder auf dem Bildschirm dargestellt werden:

```
GSHAPE A$,X,Y,M
```

A$ ist der uns nun schon bekannte Speicherstring, der durch SSHAPE definiert wurde. Es kann jede Stringvariable oder ein String-Array-Element verwendet werden.

X,Y sind die Koordinaten im aktuellen System, an die die linke obere Ecke unseres Shapes gelegt werden soll.

Interessant ist die Angabe von M. Hier dreht es sich um einen Darstellungsmodus, von dem es hier 4 Möglichkeiten gibt:

M=0. Unser Shape wird genauso abgebildet, wie es definiert wurde. Schon an dieser Stelle auf dem Bildschirm vorhandene Objekte werden überdeckt. Dasselbe ergibt sich, wenn wir diesen Parameter einfach weglassen.

M=1. Das Shape wird invertiert abgebildet. Auch hier überdeckt das Shape vorhandene Objekte.

M=2. OR-Modus: Bildpunkte werden sichtbar, wenn sie zum Shape oder zu vorhandenen Objekten gehören. Beide Bilder überlagern sich so.

M=3. AND-Modus: Bildpunkte werden nur dann gesetzt, wenn sie sowohl zum Shape als auch zum Bildschirmobjekt gehören.

M=4. EOR- (oder XOR-) Modus: Bildpunkte werden nur dann gesetzt, wenn das entsprechende Bit für Shape und Bildschirmobjekt ungleich ist.

Die Modi 2, 3, und 4 sind etwas schwer vorstellbar. Zur Verdeutlichung finden Sie hier ein kleines Demonstrationsprogramm namens GSHAPE-MODI.

Hier wird zuerst ein Shape definiert, dann der gesamte Bildschirm quergestreift und schließlich das soeben erstellte Shape in allen 5 Modi darauf dargestellt. Die Wirkung ist auf diese Weise ganz gut zu erkennen, besonders bei den Modi 2, 3 und 4.

Der Modus ist aufgrund der EOR-Behandlung interessant: Wird auf dieselbe Stelle noch einmal das Shape im Modus 4 gezeichnet, dann verschwindet es ganz, und der Bildschirmhintergrund ist wieder vorhanden. Will man Shapes bewegen, ohne Bildschirmobjekte zu zerstören, dann ist diese Darstellungsweise ganz gut geeignet.

Die Frage, wie viele Shapes man definieren und auch darstellen könne, erübrigt sich fast: beliebig viele, solange der Speicherplatz für die Strings reicht. Man könnte sich also eine eigene Bibliothek von Hochauflösungs- oder Multicolorobjekten zulegen und diese dann bei Bedarf abrufen. Sollte für ein Objekt der String zu kurz sein, können mehrere Shapes gekoppelt werden. Sie erkennen schon: Auch hier sind die Möglichkeiten sehr breit.

Zusammenspiel von SSHAPE und GSHAPE

Eine Frage haben wir noch offengelassen, die eine recht verlockende Konsequenz in sich birgt: Wenn man mittels der SCALE-Anweisung die Systeme bei SSHAPE (also beim Aufbau des Shape) und bei GSHAPE (bei der Abbildung) unterschiedlich wählt, kann man dann Shapes verkleinern oder vergrößern?

Die Antwort ist leider nein. Bei GSHAPE werden nur die Koordinaten eines (des linken oberen) Eckpunktes angegeben. Somit hat ein unterschiedliches Koordinatensystem lediglich auf den Darstellungsort eine Wirkung, nicht aber auf die Shape-Größe.

Im Gegensatz zu den Sprites der anderen Commodore-Computer ist das Bewegen von Shapes eine langweilige Angelegenheit. Jedem Shape-Aufbau durch GSHAPE kann man ganz geruhsam zusehen. Zwar ist es durch Sequenzen wie.

```
FOR  = 0 TO 200
GSHAPE A$,I,I,4
GSHAPE A$,I,I,4
NEXT I
```

möglich, Shapes ohne Schaden für den Bildschirminhalt über den Sichtbereich ziehen zu lassen. Das Ganze ähnelt aber bei weitem nicht der Sprite-Bewegung.

Der C16 wird oft unterschätzt. Zwar ist er vom Speicherplatz her - besonders bei eingeschalteter Grafik - wirklich reichlich beschränkt, von seinem Basic her ist er allerdings manchem weitaus kostspieligeren Computer hoch überlegen, was auch diese beiden kleinen Befehle zeigen.

(Heimo Ponnath/dm)

```
10 REM *** DIE GSHAPE-MODI ***
20 REM ERSTELLEN EINES SHAPE
30 COLOR 0,1: COLOR 1,6: COLOR 4,1: GRAPHIC
   1,1
40 CIRCLE 1,10,10,10,10: BOX 1,0,0,20,20: PA
   INT 1,1,1: PAINT 1,19,1: PAINT 1,1,19
50 PAINT 1,19,19: CIRCLE 1,10,10,7,7: DRAW 1
   ,10,0 TO 10,20: DRAW 1,0,10 TO 20,10
60 SSHAPE A$,0,0,20,20
70 SCNCLR
80 REM BILDSCHIRMHINTERGRUND
90 DX=5: DY=5: X=0: Y=0
100 DO WHILE X<511
110 : X=X+DX: Y=Y+DY
120 : DRAW 1,0,Y TO X,0
130 LOOP
140 REM ABBILDEN DES SHAPE
150 FOR I=0 TO 4
160 : GSHAPE A$,10+55*I,100,I
170 NEXT I
180 REM KOMMENTAR
190 COLOR 1,3
200 FOR I=0 TO 4
210 : B$=RIGHT$(STR$(I),1)
220 : CHAR 1,2+7*I,18;B$
230 NEXT I
240 CHAR 1,1,4,"DIE GSHAPE-MODI UND IHRE WIR
    KUNGEN:",1
250 CHAR 1,1,23,"BITTE EINE TASTE"
260 GET KEY A$
270 GRAPHIC 0,1
280 END
```

Listing 1. Die verschiedenen Modi beim GSHAPE-Befehl und ihre Wirkungen