C 64 Grafik Handbuch für Einsteiger

Shaffer & Shaffer

pandabooks

C64 Grafik Handbuch für Einsteiger

C64 Grafik Handbuch für Einsteiger

von Dan und Lynn Shaffer

Pandabooks, Berlin

Titel der englischsprachigen Originalausgabe:
Commodore 64 Color Graphics: A Beginner's Guide

Deutsche Übersetzung:
Carla Froitzheim und Hans Mues

ISBN 3-89058-013-0

Printed in West Germany

Inhalt

Einführung

Willkommen in der Welt der Computergrafik! Mit diesem Buch lernen Sie, Ihren **Commodore** 64 durch ein Meer von Farben zu steuern. Mit Farben Ihrer Wahl können Sie Ihre eigenen Kunstwerke erstellen!

Sie haben sicher schon gemerkt, daß das Bild auf der Rückseite mit einem **Commodore 64** produziert wurde. Das Computer-Programm, mit dem dieses Bild gezeichnet wurde, ist in den folgenden Kapiteln Schritt für Schritt erklärt. Wenn Sie dieses Buch durchgelesen haben, sind Sie sowohl in der Lage, dasselbe Bild selbst anzufertigen, als auch Programme für Ihre eigenen Bilder auf dem **Commodore 64** zu schreiben. Das Beispielprogramm und die Illustrationen in diesem Buch sind speziell für die Benutzung auf dem **Commodore 64** geschrieben. Sie lernen, BASIC-Programmierung zu benutzen, um den **Commodore 64** für das Zeichnen verschiedener Arten von Linien und Formen zu instruieren. Zusätzlich lernen Sie, wie Sie eine Zusammenstellung von Farben benutzen und wie Sie Formen in Ihren Bildern zur Bewegung veranlassen können.

Dieses Buch ist für Personen jeden Alters geeignet. "Computer-Jargon" wurde, soweit wie möglich, durch klare Umgangssprache ersetzt. Als einzige Voraussetzung benötigen Sie etwas Erfahrung im Umgang mit der Programmiersprache BASIC. Wenn Ihnen Worte wie "Programm", "Zeilennummer", "GOTO" und "RUN" geläufig sind, dürften Sie keine Probleme bei der Arbeit mit diesem Buch haben. Sollten diese Worte Ihnen jedoch den kalten Schweiß auf die Stirn treiben, müssen Sie vorher noch einige Hausaufgaben erledigen. Lesen Sie die Kapitel 2 bis 4 in Ihrem **Commodore 64** Handbuch, üben Sie ein wenig und Sie sind auf die Arbeit mit diesem Buch vorbereitet.

Was Sie in diesem Buch erwartet

Mit diesem Buch lernen Sie:

- das Zeichnen einer Hintergrundfarbe für ein Bild
- das Setzen eines Punktes
- das Zeichnen einer Linie
- das Plazieren einer Linie an eine gewünschte Stelle
- die Bildung von Formen
- das Plazieren einer Form an beliebiger Stelle auf dem Bildschirm

- die Beeinflussung von Farben
- das Zeichnen und Bewegen von kleinen Objekten

Sämtliche Techniken bilden einen notwendigen Bestandteil jeder Bildkomposition auf Ihrem Bildschirm. Da jede notwendig ist, haben wir dieses Buch in einer bestimmten Weise aufgebaut. Wenn Sie am Ende des Kapitel 6 angelangt sind, verfügen Sie über einen "Werkzeugkasten", der die Grafiktechniken als einzelne "Werkzeuge" enthält. Dieser Werkzeugkasten kann leicht von Programm zu Programm transportiert werden. Sie wollen eine Linie zeichnen? Kein Problem. Nehmen Sie einfach Ihr "Linie zeichnen"-Werkzeug, geben Sie die Stelle für die Linie an und der Fall ist erledigt! (Sie werden dies noch besser verstehen, wenn das erste Werkzeug in Kapitel 2 erläutert wird.)

Wir konzentrieren uns darauf, **wie** ein Bild auf dem **Commodore 64** gezeichnet wird. Oft ist die Kenntnis des **warum** nicht wichtig, um das Bild zu erstellen. Denken Sie an die Benutzung Ihres Radios. Es interessiert Sie wahrscheinlich nicht, **warum**, sondern **wie** es funktioniert (wo der Ein/Ausschalter sitzt). In jedem Kapitel ist das "warum", welches für das Verständnis nicht wichtig ist, vom Rest des Textes getrennt und in einen separaten Kasten gesetzt. Sie können die technischen Beschreibungen lesen oder überspringen, ganz wie Sie wollen. Überspringen Sie diese, hindert Sie das in keiner Weise daran, zu lernen, Ihre eigenen grafischen Entwürfe zu erstellen.

Vorbereitungen

Wenn Sie mit diesem Buch arbeiten, benötigen Sie folgende Geräte:

1. 1 Diskettenlaufwerk (mit einer leeren Diskette) oder einen Cassettenrecorder für den **Commodore 64** (mit einem leeren Band);
2. Eine **Commodore 64** Tastatur;
3. Einen Farbmonitor oder ein Farbfernsehgerät und
4. Zeichenpapier, um Ihre eigenen Entwürfe auszuarbeiten.

Wenn Sie mit diesem Buch arbeiten, sollten Sie immer am Computer sitzen. Sie benötigen einen Monitor (z.Bsp. ein Fernsehgerät), der korrekt an den **Commodore** angeschlossen ist. Instruktionen, wie Sie Ihre Geräte anschließen, finden Sie im Handbuch zu Ihrem Computer. Dort wird auch erläutert, wie Sie einen Cassettenrecorder an den **Commodore** anschließen können.

Wenn Sie ein Diskettenlaufwerk besitzen, schließen Sie dies am **Commodore** an. Falls nötig, sehen Sie im Handbuch zu Ihrem Diskettenlaufwerk nach, wie es korrekt angeschlossen wird.

Sobald Ihre Geräte einsatzbereit sind, gehen Sie folgendermaßen vor:

Schalten Sie

1. das Laufwerk ein, falls Sie eines benutzen. (Cassettenrecorder sind automatisch eingeschaltet, wenn Sie am Computer angeschlossen sind.)
2. den **Commodore 64** ein.
3. den Monitor oder den Fernseher ein.

Sie sehen " **** COMMODORE 64 BASIC V2 **** " oben auf dem Bildschirm und darunter eine Zeile, die den Speicher Ihres Computers beschreibt. (Machen Sie sich keine Gedanken, wenn Sie nicht wissen, was "Speicher" bedeutet.) Außerdem sehen Sie READY, gefolgt von einem blinkenden Quadrat. Dieses blinkende Quadrat wird **Cursor** genannt, der zur Markierung einer Stelle auf dem Bildschirm dient. Jedesmal, wenn Sie einen Buchstaben oder ein Symbol eingeben, wird dieses Zeichen an der Stelle angezeigt, an der der Cursor steht. Der Cursor wandert gleichzeitig eine Stelle nach rechts, um die Stelle für das nächste Zeichen zu markieren. Sie werden den Cursor noch oft sehen, wenn Sie das Programm eingeben, das Ihr Schiff zeichnet.

Benutzung dieses Buches

Ab Kapitel 2 gehen Sie am besten folgendermaßen vor:

1. Laden Sie das Programm aus dem vorherigen Kapitel in den Computer.
2. Lesen Sie die Einführung in das neue Kapitel.
3. Geben Sie die erforderlichen BASIC Zeilen ein.
4. Starten Sie das Programm mit RUN und beobachten Sie, was auf dem Bildschirm passiert.
5. Verschaffen Sie sich einen Überblick über das Programm mit LIST und korrigieren Sie Schreibfehler.
6. Lesen Sie die Kapitelseiten durch und gehen Sie nach den Erklärungen vor.
7. Speichern Sie am Ende eines Abschnitts oder Kapitels Ihr Programm mit SAVE unter einem neuen Namen ab.

Wenn Sie jedes Kapitel durcharbeiten, entwickeln Sie ein besseres Verständnis dafür, wie ein Bild mit dem Computer gezeichnet wird.

Wir schlagen vor, daß Sie, während Sie ein Kapitel lesen, das Programm mehrmals laufen lassen, um zu sehen, wie der Computer mit dem Programm die Bildanzeige auf Ihrem Monitor erzeugt. Die Kapitel sind so angelegt, daß jedes auf das vorherige aufbaut. Auf diese Weise gelangen Sie Schritt für Schritt zum endgültigen Bild.

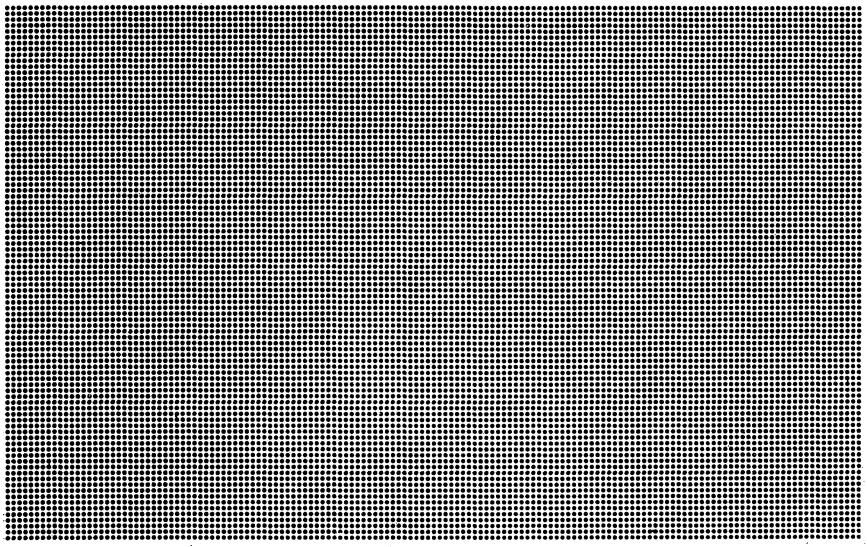
Der Anhang des Buches bietet Ihnen zusätzliche Informationen über **Commodore 64** Grafik. Außerdem finden Sie dort eine Anleitung zur Fehlersuche, die Ihnen helfen soll, Programmfehler zu lokalisieren, und ein kommentiertes Literaturverzeichnis zu Schriften über den **Commodore 64.** Kapitel 6 behandelt den Inhalt des Anhangs noch ausführlicher.

Einführung in die hochauflösende Grafik

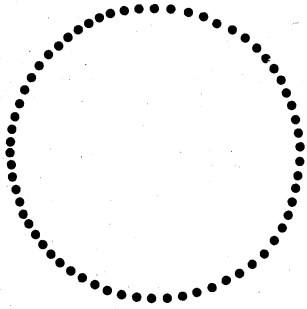
Die meisten der heute angebotenen Software-Produkte (seien es Computerspiele, Grafikprogramme, Lehrmaterialien o.ä.) benutzen hochentwickelte farbige Bilder. Computerbilder wie diese bestehen aus tausenden von winzigen Punkten farbigen Lichtes. Diese farbigen Punkte werden **Pixels** genannt. Jedes Bild des **Commodore 64** besteht aus insgesamt 64.000 Pixels, 320 in jeder Zeile und 200 in jeder Spalte. Diese Pixels sind sehr klein und liegen dicht beieinander. Deshalb ergeben sie ein geschlossenes Bild, wenn Sie sich in einiger Entfernung vom Monitor oder Fernseher befinden.

Die Gesamtzahl der Pixels, die ein Computer zur Anzeige eines Bildes auf dem Bildschirm benutzt, wird **Auflösung** genannt. Die Qualität eines Computerbildes variiert von Gerät zu Gerät, da einige Computerbilder mit einer Auflösung von 64.000 Pixels anzeigen können, andere hingegen nur mit erheblich weniger, beispielsweise 1.600 Pixels pro Bild. Je höher die Anzahl der Pixels ist, desto höher ist die Auflösung. Dies wird **Hohe Auflösung** genannt. Alle Grafikprogramme in diesem Buch werden in hoher Auflösung angezeigt. Ihr hochauflösender Bildschirm erscheint mit Reihen und Spalten von Pixels wie in folgendem Bild:

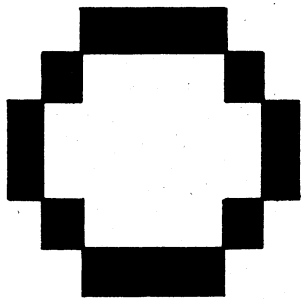
Hochauflösender Bildschirm



Bilder auf einem Bildschirm mit hoher Auflösung werden **hochauflösende Grafiken** genannt. Die Auflösung eines Bildschirms spielt eine große Rolle bei der Erstellung von Bildern: je höher die Auflösung ist, desto mehr Details kann das Bild haben. Wenn Sie beispielsweise einen Kreis in hoher Auflösung zeichnen, sieht er folgendermaßen aus:



während Ihr Kreis, wenn Sie mit einem anderen Computersystem mit niedriger Auflösung arbeiten, eher wie dieser aussieht:



Ein Fernseher oder Monitor mit niedriger Auflösung hat weniger Pixels. Um diesen mit weniger Pixels auszufüllen, muß der Computer größere Pixels verwenden. Das läßt Objekte, z.Bsp. Kreise, eckig und plump im Vergleich zum ebenmäßigen Aussehen der hochauflösenden Bilder erscheinen.

Ob Sie nun Tabellenzeichnungen für Geschäftszwecke oder eine Karte für den Geographieunterricht erstellen, sie werden die Arbeit mit der hochauflösenden Grafik auf Ihrem **Commodore 64** zu schätzen lernen.

Einführung in die C-64 Tastatur

Wenn Sie mit Ihrem **Commodore 64** schon gearbeitet haben, wissen Sie wahrscheinlich, wie einige der speziellen Tasten eingesetzt werden. Einige Tasten ermöglichen beispielsweise die Änderung der Farben der eingegebenen Buchstaben und Symbole.

Um zu sehen, wie dies funktioniert, beginnen wir mit der Änderung der Schriftfarbe in gelb. Betätigen Sie die CTRL-Taste (links auf der Tastatur) und **gleichzeitig** die Taste mit der Ziffer 8 (obere Tastenreihe). Dann lassen Sie beide Tasten los.

Beachten Sie: Wenn wir "betätigen Sie CTRL" angeben, meinen wir damit, daß Sie die Taste mit der Bezeichnung CTRL und nicht, daß Sie jeden Buchstaben (C, T, R, und L) drücken sollen.

Um zu sehen, was passiert, geben Sie Ihren Namen ein. Er erscheint in Gelb. Betätigen Sie die CTRL-Taste und die Taste 3. Lassen Sie die Tasten los und geben Sie ein weiteres Wort ein. Welche

Farbe haben die Buchstaben jetzt? Probieren Sie einige Farben aus (CTRL 6, loslassen) und geben Sie einige Worte und Zeichen ein. Stellen Sie Ihren Bildschirm so ein, daß Sie ein deutliches Bild haben.

Dies ist nur eine Möglichkeit, den Computer bei der Benutzung der Farben zu steuern. Ein weiterer, einzigartiger Bestandteil des **Commodore 64** ist die spezielle "Commodoretaste" (C =). Mit Hilfe dieser Taste verfügen Sie über einen zweiten Satz Zeichenfarben. Die Taste befindet sich links unten auf der Tastatur. Betätigen Sie die Tasten C = und 5 gleichzeitig. Lassen Sie beide Tasten los und geben Sie einige weitere Zeichen ein. Sie benutzen jetzt den zweiten Farbensatz. Betätigen Sie C = 6 und geben Sie das Wort Commodore ein. Nehmen Sie sich etwas Zeit und probieren Sie die anderen Farbmöglichkeiten aus.

Unten sehen Sie eine Übersicht der 16 verschiedenen Farben. Für eine Farbe des ersten Satzes (oben), wie z.B. Rot, betätigen Sie CTRL 3. Für eine Farbe des zweiten Satzes (unten), wie z.B. Hellgrün, betätigen Sie C = 6.

Übersicht der Zeichenfarben

erster Satz

Schwarz Weiss Rot Cyan Purpur Grün Blau Gelb

CTRL

1 2 3 4 5 6 7 8

C =

Orange Braun Hell Rot Grau 1 Grau 2 Hell Grün Hell Blau Grau 3

zweiter Satz

Wenn Sie auf Ihre Tastatur sehen, finden Sie eine Reihe von Grafikzeichen auf der Vorderseite der Buchstabentasten. Auf der vorderen Seite der N-Taste befinden sich beispielsweise zwei grafische Zeichen: ◪ auf der rechten, ◧ auf linken Seite. Das

rechte Zeichen wird mit der SHIFT-Taste und der Buchstabentaste mit dem gewünschten Zeichen auf der Vorderseite angezeigt. Das linke Zeichen erhalten Sie mit C = und der entsprechenden Buchstabentaste. Geben Sie nun diese Grafikzeichen in verschiedenen Farben ein. Obwohl die Zeichen für einige grafische Beispiele ganz nützlich sind, reichen sie für das Grafikprogramm in diesem Buch nicht aus, deshalb werden wir sie nicht verwenden.

Um den Bildschirm zu löschen, betätigen Sie die SHIFT- und gleichzeitig die CLR/HOME-Taste (CLEAR/HOME) oben rechts auf der Tastatur. Der Bildschirm wird gelöscht und der Cursor in die obere linke Ecke des Bildschirms bewegt. Diese Stelle wird HOME genannt.

Eine andere Taste, die Sie oft benutzen müssen, ist die RUN/STOP-Taste (links auf der Tastatur). Wenn Sie ein Programm laufen lassen, wie im nächsten Kapitel, wird das Programm mit der RUN/STOP-Taste angehalten. Falls ein Programm beispielsweise anders arbeitet, als Sie es erwarten, betätigen Sie diese Taste. Sie können dann zurückgehen und das Programm auf Eingabefehler hin überprüfen. Wir erinnern Sie später noch einmal an diese Taste, wenn Sie Ihr erstes Programm starten.

Es besteht aber auch die Möglichkeit, daß Sie mit der Taste RUN/STOP nicht das Programm-Listing erhalten. Wenn Sie ein Programm falsch eingegeben haben und die RUN/STOP Taste zeigt keine Wirkung, betätigen Sie diese gleichzeitig mit der RESTORE-Taste. Dann müßten Sie Ihr Programm-Listing eigentlich erhalten. Wenn nicht, hilft nur noch eines: den Computer aus- und wieder einschalten. Diese Methode hat allerdings den Nachteil, daß jeder Programmteil, der noch nicht auf Diskette oder Band gespeichert wurde, vom Computer "vergessen" (gelöscht) wird.

Unten sehen Sie eine Übersicht der speziellen Tasten, die wir bisher behandelt haben:

Taste	Verwendung
CTRL 8	bestimmt die Farbe gelb für die eingegebenen Zeichen
C = 6	bestimmt die Farbe hellgrün für die eingegebenen Zeichen
SHIFT-Buchstabe	schreibt das Grafikzeichen, das sich rechts auf Taste der Vorderseite der Buchstabentaste befindet

C = Buchstabentaste
: schreibt das Grafikzeichen, das sich links auf der Vorderseite der Buchstabentaste befindet

SHIFT CLR/HOME
: löscht den Bildschirm und bewegt den Cursor in die Home-Position

RUN/STOP
: hält das laufende Programm an, Sie erhalten wieder Textanzeige; der Text, der vor dem Start des Programms auf dem Bildschirm angezeigt wurde, wird nun erneut angezeigt, ebenso Syntaxfehler, die während der Programmausführung gefunden wurden.

RUN/STOP RESTORE
: hält das laufende Programm an, sie erhalten wieder den Textbildschirm; der Textbildschirm wird gelöscht und der Cursor in die Home-Position bewegt.

Diese und andere Tasten werden in den folgenden Kapiteln benutzt, um das Programm einzugeben. Bevor Sie zu Kapitel 1 übergehen, nehmen Sie sich etwas Zeit, Ihre Tastatur zu betrachten. Sie sollten sich die Positionen aller Tasten, die Sie vorher noch nicht gesehen haben, merken. Achten Sie besonders auf die RETURN-Taste, da Sie diese mit am häufigsten benutzen werden.

Wenn Ihnen das Eintippen zu mühsam ist: Zu diesem Buch gibt es eine Begleitdiskette, auf der alle Programme enthalten sind!
Sie erhalten die Diskette (Format Commodore 1541) in Ihrer Buchhandlung oder gegen Voreinsendung von DM 30,-- (V-Scheck) bei

Pandabooks
Bismarckstr. 67
D-1000 Berlin 12

Kapitel 1
Anfertigung eines Programms

Jedes Kapitel dieses Buches behandelt einen Teil des Programms, mit dem Ihr Schiff gezeichnet wird. Ein "Programm" ist eine Liste von Anweisungen, die dem Computer mitteilen, was er zu tun hat. Jede Anweisung (oder Zeile) in einem Programm erhält eine Nummer. Die Zeilennummern geben dem Computer die Reihenfolge der Anweisungen für die Ausführung an. Zeilennummern können bei 0 beginnen und bis 63999 fortgeführt werden.

Die Anweisungen innerhalb einer Zeile bestehen aus verschiedenen Buchstaben und Symbolen. Es gibt viele verschiedene Möglichkeiten, Anweisungen an den Computer zu erteilen. Jede Methode wird **Programmiersprache** genannt. Sie werden eine Sprache namens BASIC verwenden, mit der Sie lernen, den **Commodore 64** zu instruieren, Linien und Formen zu zeichnen, Farben zu ändern und sogar Objekte zu bewegen.

Das Programm dieses Kapitels hat zwei Ziele:

1. die Struktur des Programms in seiner endgültigen Form zu bestimmen
2. Sie mit "Unterroutinen" völlig vertraut zu machen.

Generell kann man ein Programm in zwei Abschnitte teilen: die Hauptroutine und die Unterroutinen. Unterroutinen kann man sich als "Werkzeuge" (TOOLS) vorstellen, die die Hauptroutine für verschiedene Aufgaben benutzt. Denken Sie an die Werkzeuge, die zum Aussäen von Pflanzen benutzt werden: eine Hacke, ein Spaten oder eine Schaufel. Jedes dieser Werkzeuge erfüllt eine spezifische Aufgabe, wobei die Hacke einem anderen Zweck dient als der Spaten. Wenn ein Werkzeug nicht mehr benötigt wird, legt man es beiseite, bis es wieder gebraucht wird. In der Zwischenzeit erledigt der Gärtner die eigentliche Aufgabe, das Aussäen.

Unterroutinen werden in ähnlicher Weise an bestimmten Stellen eingesetzt, und nur dann genutzt, wenn sie von der Hauptroutine aufgerufen werden. Nach Erledigung ihrer Aufgabe werden sie "beiseite gelegt", während der Computer zur Hauptroutine zurückkehrt. Aufgrund ihrer häufigen Verwendung sind Unterroutinen so zusammengefaßt, daß sie für den jeweiligen Einsatz leicht aufzufinden sind. Da Unterroutinen häufig benutzt werden, sind sie zusammengefaßt.

Stellen Sie sich den Computer als den Gärtner vor. Er trägt die Verantwortung dafür, daß die Hauptroutine (das "Einsetzen der Pflanzensamen") erledigt wird, indem er die zur Verfügung stehenden Werkzeuge benutzt. Um eine Unterroutine während der Hauptroutine "aufzunehmen", wird ein spezieller Programmiercode verwendet: GOSUB. Auf GOSUB folgt immer eine Zahl, wie GOSUB 20. Wenn der Computer auf einen derartigen Befehl stößt, geht (GO) er direkt zu einer Unter-(SUB)routine. Die Zahl gibt dem Computer die Zeile an, in der er den Anfang der Unterroutine findet. Der Computer unterbricht die Ausführung der Hauptroutinen-Befehle und beginnt die Unterroutine durchzuführen. Um dem Computer mitzuteilen, an welcher Stelle die Aufgabe der Unterroutine erledigt ist, wird ein anderer Code verwendet: RETURN. RETURN bestimmt dem Computer, die Benutzung der Unterroutine zu beenden und zur Hauptroutine zurückzukehren.

An welcher Stelle kehrt der Computer in die Hauptroutine zurück? Hierin liegt eine bemerkenswerte Eigenschaft der Unterroutinen: Sobald der Computer nämlich die Unterroutine beendet hat, kehrt er an genau dieselbe Stelle in der Hauptroutine zurück, von der aus er gestartet war.

Nehmen wir als Beispiel folgenden Ausschnitt aus einem Programmteil:

```
1100 GOSUB 20
1101 PRINT "SUBROUTINE COMPLETED"
```

Wenn der Computer Zeile 1100 erreicht, geht er sofort über zu Zeile 20, um die dort befindliche Unterroutine auszuführen. Gelangt er zu einer RETURN-Anweisung, kehrt er an die Stelle direkt hinter GOSUB 20 zurück. Das ist in diesem Fall die Zeile 1101, die dem Computer angibt, die Worte "SUBROUTINE COMPLETED" anzuzeigen. Die Befehle GOSUB und RETURN ermöglichen das "Aufnehmen" eines Unterroutinen-Werkzeugs, seine Benutzung und die Rückkehr an dieselbe Stelle in der Hauptroutine.

Wie strukturiert man nun ein Programm? Da Unterroutinen von der Hauptroutine getrennt sein sollen, um einen einfachen Zugriff zu gewährleisten, erhalten sie ihre eigenen Zeilennummern. Normalerweise werden für diesen Zweck 1.000 bis 2.000 Zeilennummern nur für die Unterroutinen reserviert. Das endgültige Programm in diesem Buch enthält 23 Unterroutinen. Um sicherzugehen, daß genügend Platz bleibt, um sie alle einzugeben, reservieren wir 1.000 Zeilennummern nur für die Unterroutinen. Um genügend Platz für die Hauptroutine zu lassen, wird das Programm folgendermaßen aufgebaut:

1. die Zeilen 1 bis 999 werden nur für die Unterroutinen verwendet
2. die Zeilen ab 1000 werden nur für die Hauptroutine verwendet

Dieser Aufbau erlaubt es Ihnen, die Hauptroutine in der notwendigen Größe zu erstellen. Denken Sie daran, daß die Hauptroutine für den Hauptteil der Arbeit verantwortlich ist, weshalb sie mehr Zeilen für Anweisungen benötigt.

Eingabe des Programms

Damit Sie sehen, wie dies funktioniert, geben Sie das erste, für den Anfang einfache Programm ein. Dieses Programm läßt den Computer eine Meldung auf dem Bildschirm anzeigen, wenn eine von 3 "speziellen" Tasten gedrückt wird. Wenn eine andere Taste benutzt wird, passiert nichts. Klingt einfach? Ist es auch.

Zuerst betätigen Sie CTRL und 2 gleichzeitig. Sie erinnern sich? Damit ändern Sie die Schriftfarbe in weiß, die auf dem Bildschirm besser zu erkennen ist als blau.

Nun geben Sie das unten aufgeführte Programm ein. Vergessen Sie die Zeilennummern nicht. Sie teilen dem Computer mit, in welcher Reihenfolge die Anweisungen auszuführen sind. Geben Sie das Programm sorgfältig ein, lassen Sie nichts aus und fügen Sie nichts hinzu. Eine kleine Änderung kann schon bewirken, daß das Programm nicht mehr korrekt abläuft. Wir haben das Programm eingerückt, damit es gut zu erkennen ist. Rücken Sie es bei der Eingabe **nicht** ein. Beginnen Sie am linken Rand, geben Sie Zeile 1 ein und betätigen Sie RETURN. Fahren Sie fort mit Zeilennummer 20.

```
1 GOTO 1000
20 REM:::::::::GRAPHICS
21 PRINT"SUBROUTINE G WORKS"
24 RETURN
30 REM:::::::::TEXT
31 PRINT"SUBROUTINE T WORKS"
34 RETURN
40 REM:::::::::COLORS
41 PRINT"SUBROUTINE C WORKS"
44 RETURN
1000 REM::::::::::::::::::
1001 REM       MAIN ROUTINE
1002 REM::::::::::::::::::
1100 GET A$
1110 IF A$ = "G" THEN GOSUB 20
```

```
1120 IF A$ = "T" THEN GOSUB 30
1130 IF A$ = "C" THEN GOSUB 40
1140 GOTO 1100
```

Fehlerberichtigung:

Wenn Sie einen Schreibfehler machen, **bevor** Sie RETURN drücken, betätigen Sie die INST/DELTaste (oben rechts auf der Tastatur), um zurückzugehen und Zeichen zu löschen.

Wenn Sie einen Fehler machen, **nachdem** Sie RETURN gedrückt haben, geben Sie die ganze Zeile (einschließlich der Zeilennummer) neu ein. Es sieht so aus, als ob Sie zwei Zeilen mit derselben Nummer hätten, aber wenn Sie LIST eingeben und RETURN betätigen, sehen Sie, daß die korrekte Zeile in das Programm eingefügt worden ist.

Wenn Sie eine ganze Zeile löschen wollen (z.B. eine mit einer falschen Zeilennummer), geben Sie die Nummer der zu löschenden Zeile nochmals ein und betätigen Sie RETURN. Geben Sie LIST ein, drücken Sie RETURN und Sie sehen, daß die Zeile aus Ihrem Programm gelöscht worden ist.

Sehen Sie sich das Programm an. Es enthält drei Unterroutinen. Jede befindet sich in der Gruppe der Zeilennummern 1 bis 999. Die Hauptroutine beginnt bei Zeilennummer 1000. Sie enthält die Befehle GOSUB, um die Unterroutinen zu benutzen. Diese wiederum beinhalten den Befehl RETURN, um zur Hauptroutine zurückzugelangen.

Start des Programms

Wenn Sie das Programm auf Ihren Bildschirm geschrieben haben, können Sie es mit dem Befehl "RUN" starten. RUN ist der Befehl, den Sie eingeben, wenn der Computer mit Ihrem Programm arbeiten soll. Er teilt dem Computer mit, bei der niedrigsten Zeilennummer im Programm beginnend (in Ihrem Programm ist es Zeile 1), die Anweisungen des Programms zu erfüllen. Versuchen Sie es. Geben Sie das Wort RUN ein und betätigen Sie RETURN.

Sie werden noch sehen, warum das Programm sofort arbeitet. Jetzt sagen wir Ihnen erst einmal, was Sie erwarten kann und was Sie tun müssen. Wenn Ihr Programm nicht im beschriebenen Sinne arbeitet, betätigen Sie die RUN/STOP-Taste. Dann geben Sie LIST ein und betätigen RETURN. Wenn Sie dann Ihr Programm wieder auf dem Bildschirm sehen, sollten Sie es auf Schreibfehler hin überprüfen. Häufige Fehler sind ausgelassene Zeilen, falsche Zeilennummern, ausgelassene Anführungszeichen, kleines L (l) anstelle

von eins (1) und O anstelle von Null (0). Geben Sie fehlerhafte Zeilen neu ein und starten Sie (RUN) das Programm noch einmal.

Wenn das Programm läuft, stellen Sie als erstes fest, daß der Cursor verschwindet. Sonst ändert sich nichts. Der Computer erwartet, daß Sie eine Taste betätigen. Erinnern Sie sich, er wartet auf eine von 3 speziellen Tasten. Drücken Sie die Taste 5. Wenn Sie das Programm richtig eingegeben haben, passiert nichts. Versuchen Sie es mit H. Wieder nichts. Nun probieren Sie T.

Der Computer antwortet mit der Meldung "SUBROUTINE T WORKS". Betätigen Sie G und C. G, T und C sind die speziellen Tasten, die den Computer veranlassen, eine Meldung zu schreiben. Wie Sie bereits wissen, sind das die Buchstaben, die Sie in Anführungszeichen in Ihr Programm eingegeben haben. Probieren Sie noch einige Male diese und andere Tasten aus. Sie finden schnell heraus, daß Sie nur mit diesen drei Tasten eine Meldung erhalten.

Um herauszufinden, wie dies funktioniert, betätigen Sie die RUN/STOP-Taste, um das Programm zu beenden. Auf dem Bildschirm erscheinen wieder "READY" und der Cursor. Geben Sie LIST ein und betätigen Sie RETURN, um das Programm wieder zu sehen.

Aufgliederung des Programms

Die erste Zeile des Programms befiehlt dem Computer, die Unterroutinen zu überspringen und zu Zeile 1000 zu gehen (GOTO). Wenn diese Zeile ausgelassen würde, könnten wir die Unterroutinen nicht vor die Hauptroutine setzen.

Zeilen 1000 bis 1002 enthalten "Bemerkungen". Solche Zeilen beginnen mit dem Wort REM und werden vom Computer **immer** ignoriert. Ihre Funktion besteht darin, verschiedene Abschnitte des Programms zu kennzeichnen oder zu betiteln. Diese Zeilen kennzeichnen demnach die HAUPTROUTINE in Ihrem Programm. REM-Zeilen dienen nur dazu, Sie daran zu erinnern, welche Bedeutung die verschiedenen Teile des Programms haben.

Als nächstes gelangt der Computer zu Zeile 1100, die GET A$ enthält. Damit befehlen Sie dem Computer, sich A$ zu beschaffen und "herauszufinden", um was es sich handelt. Der Computer sieht zur Tastatur und wartet auf die Betätigung einer einzelnen Taste. Jede betätigte Taste wird vom Computer wahrgenommen und in die "Leerstelle" namens A$ eingesetzt. Stellen Sie sich A$ als Platzhalter vor, der schließlich mit dem von Ihnen gewählten, in die Tastatur eingegebenen Zeichen (Buchstabe, Zahl oder Symbol) ausgefüllt wird.

Sobald Sie ein Zeichen eingegeben haben, weiß der Computer, was A$ ist und kann zur nächsten Zeilennummer weitergehen.

Die Zeilen 1110 bis 1130 teilen dem Computer, da er jetzt A$ identifizieren kann, mit, was er tun zu hat. Diese Art von Zeilen werden IF...THEN-Befehle genannt. Sie befehlen dem Computer "WENN ... wahr ist, DANN führe ... aus. Die Zeile 1110 lautet:

```
IF A$ = "G" THEN GOSUB 20
```

Der Computer liest dies als "WENN A$ der Buchstabe G ist, DANN gehe (GO) zur Unter-(SUB)routine in Zeile 20". Die Zeilen 1120 und 1130 teilen dem Computer mit, was er zu tun hat, wenn A$ T oder C ist. Sie dirigieren den Computer ebenfalls zu einer Unterroutine. Wenn A$ nicht G, T oder C ist, geht der Computer weiter zu Zeile 1140. Diese dirigiert ihn zurück zu Zeile 1100, damit es sich erneut A$ beschafft. Die nächste betätigte Taste entspricht dann A$ und der Computer "vergißt", daß vorher eine andere betätigte Taste A$ entsprach.

Zum Schluß sehen wir uns die Unterroutinen an. Angenommen, das Programm läuft und der Computer gelangt zu Zeile 1100 GET A$. Sie drücken die Taste T. Dann geht er weiter zu Zeile 1110. Da A$ nicht G entspricht, überspringt der Computer den Befehl in dieser Zeile und fährt fort. Er erreicht Zeile 1120. Da A$ T entspricht, folgt der Computer den Instruktionen in dieser Zeile, zur Unterroutine in Zeile 30 zu gehen.

Zeile 30 enthält wieder das Wort REM, deshalb ignoriert der Computer die Zeile. Zeile 31 enthält eine PRINT-Anweisung. Sie besagt, daß "SUBROUTINE T WORKS" geschrieben werden soll. Alle Ausdrücke in Anführungszeichen hinter dem Wort PRINT werden auf dem Bildschirm angezeigt. Aus diesem Grund schreibt Ihr Computer jedesmal SUBROUTINE T WORKS, wenn Sie die Taste T betätigt haben. Sobald die Meldung ausgegeben ist, liest der Computer die Zeile 34. Diese enthält den notwendigen Befehl RETURN für die Rückkehr zur Hauptroutine. Der Computer weiß, daß die Aufgabe der Unterroutine beendet ist und kehrt zur Hauptroutine zurück.

Er geht, wie schon beschrieben, an die Stelle zurück, an der er die Hauptroutine verlassen hat, in diesem Fall zu Zeile 1130. A$ entspricht immer noch T, deshalb wird die Zeile 1130 gelesen und übergangen. Zeile 1140 befiehlt GOTO (Zeile) 1100 und der Vorgang beginnt von neuem.

Es ist wichtig, daß Sie diese allgemeinen Grundlagen von BASIC verstehen, da Sie viele Unterroutinen, GOTO-, GET- und PRINT-Befehl in den folgenden Kapiteln benutzen werden.

Starten Sie Ihr Programm noch einmal (geben Sie RUN ein und betätigen Sie RETURN). Verfolgen Sie währenddessen obige Programmaufgliederung und achten Sie darauf, wie der Computer mit Ihrem Programm arbeitet. Wenn Sie Schwierigkeiten haben, überprüfen Sie das Programm-Listing auf Schreibfehler. Korrigieren Sie Zeilen, indem Sie sie neu eingeben.

Wenn Sie dies beendet haben, betätigen Sie RUN/STOP, um das Programm abzubrechen. Geben Sie LIST ein und drücken Sie RETURN, um das Programm wieder zu sehen. Ihre letzte Aufgabe in diesem Kapitel ist das **Sichern** Ihres Programmes.

Sichern (SAVE) eines Programms

Sobald ein Programm korrekt läuft, sollten Sie es auf Ihrer Diskette oder Cassette abspeichern (sichern). Was bedeutet "Sichern" eines Programms, und warum wird dies notwendig?

Wenn Sie ein Programm eingeben, wird es im "Speicher" des Computers "gelagert". Der Speicher des Computers ist ein kurzfristiger Lagerplatz innerhalb des Computers, der die eingegebenen Informationen aufnimmt. Er ist ein kurzfristiger Lagerplatz, da die Informationen darin ausgelöscht werden, sobald der Computer ausgeschaltet wird. Dies trifft auch auf die eingegebenen Programme zu. Ihr Diskettenlaufwerk oder Cassettenrecorder löst dieses Problem durch Herstellung einer dauerhaften "Aufzeichnung" Ihres Programms. Jede Aufzeichnung kann dann beliebig oft abgespielt werden.

Es gibt verschiedene Möglichkeiten ein Programm zu **sichern**, je nachdem, ob Sie ein Diskettenlaufwerk oder einen Cassettenrecorder benutzen. Bevor Sie nun beginnen, ein Programm abzuspeichern, sollten Sie Ihr Laufwerk oder Ihren Recorder möglichst weit entfernt vom Monitor plazieren. Der Monitor könnte sonst den Abspeichervorgang stören, so daß Ihre Programme nicht korrekt gesichert werden.

Wenn Sie ein Diskettenlaufwerk benutzen, lesen Sie den nächsten Abschnitt "Sichern (SAVE) auf einem Diskettenlaufwerk". Benutzen Sie einen Cassettenrecorder, lesen Sie den übernächsten Abschnitt "Sichern (SAVE) auf einem Cassettenrecorder".

Sichern (SAVE) auf einem Diskettenlaufwerk

Vergewissern Sie sich zunächst, ob das Laufwerk an den **Commodore 64** angeschlossen ist. Um das Programm aus Kapitel 1 zu sichern, geben Sie ein:

```
SAVE "KAPITEL1",8
```

Sie müssen KAPITEL1 in Anführungszeichen eingeben und dürfen keine zusätzlichen Leerstellen innerhalb der Anführungszeichen einfügen. Nachdem Sie diese Befehlszeile eingegeben haben, betätigen Sie die RETURN-Taste. Diese Taste teilt dem Computer mit, daß Sie die Eingabe des Befehls beendet haben.

> Beachten Sie: Sie müssen RETURN nach jeder Befehlszeile im Programm-Listing und nach jedem Befehl drücken. Auch wenn wir Sie ab und zu an die Betätigung der RETURNTaste erinnern, müssen Sie meist selbst daran denken.

Der Computer beginnt, das Programm abzuspeichern. Sobald der Vorgang beendet ist, erlischt das rote Betriebssignal des Laufwerks.

Der Befehl, mit dem Ihr Programm auf der Diskette gesichert wird, lautet SAVE. Diesem Befehl folgt immer der Dateiname (Titel) des Programms in **Anführungszeichen.** Sie können Ihr Programm mit jedem gewünschten Namen bezeichnen, vorausgesetzt, der Name besteht aus höchstens 16 Zeichen. Hinter den Dateinamen setzen Sie ein Komma und die Zahl 8, die dem Computer mitteilt, daß Sie ein Diskettenlaufwerk benutzen (und nicht einen Cassettenrecorder).

Beachten Sie, daß jedes neue Programm unter einem eigenen Dateinamen abgespeichert werden muß. Es wird aber auch vorkommen, daß Sie ein altes Programm löschen und ein neues unter dem alten Dateinamen sichern müssen, beispielsweise, wenn Sie ein Programm aktualisieren. Um ein Programm auf Ihrer Diskette zu löschen und zu ersetzen, geben Sie ein:

```
SAVE "§0:Dateiname",8
```

"Dateiname" ist der Name des alten Programms, das Sie löschen und ersetzen wollen.

Überprüfung des Abspeichervorgangs:

Wenn Sie ein Programm sichern, sollten Sie anschließend überprüfen, ob es wirklich abgespeichert wurde. Dazu benutzen Sie einen anderen Befehl: VERIFY. Dieser Befehl kontrolliert, ob das Programm sicher auf der Diskette abgespeichert ist. Zur Überprüfung des Programms aus Kapitel1 geben Sie ein:

```
VERIFY "KAPITEL1",8
```

Sie sehen, daß der einzige Unterschied zwischen dem Abspeicher- und dem Kontrollbefehl im ersten Wort besteht (SAVE oder VERIFY). Der Teil "Kapitel1",8 wird nicht geändert. Der Computer antwortet mit SEARCHING FOR KAPITEL1 VERIFYING OK. Wenn nicht, geben Sie den Befehl VERIFY noch einmal ein. Wenn Sie immer noch nicht die richtige Antwort erhalten, befindet sich das Programm nicht auf der Diskette. Geben Sie LIST ein, um zu überprüfen, ob sich das Programm noch im Speicher befindet (wenn nicht, müssen Sie es noch einmal eingeben), daraufhin wiederholen Sie den SAVE-Befehl. In den meisten Fällen liegt der Grund für das Nichtauffinden eines Programms darin, daß sich ein Schreibfehler im Befehl VERIFY oder SAVE eingeschlichen hat.

Laden eines Programmes von der Diskette:

Da das Programm nun auf einer Diskette gesichert ist, können Sie eine Pause machen. Vergewissern Sie sich, daß das rote Betriebssignal des Laufwerks nicht mehr leuchtet, nehmen Sie die Diskette heraus und schalten Sie den Computer aus. Das Programm wird aus dem Speicher gelöscht, aber es bleibt auf der Diskette.

Wenn Sie den Computer einschalten und das Programm wieder in den Speicher holen wollen, **laden** Sie es von der Diskette. Laden eines Programms bedeutet nicht, daß es von der Diskette entfernt wird und in den Speicher gelangt, sondern es wird jeweils nur in den Speicher kopiert, während das Original auf der Diskette bleibt.

Sie benutzen den Befehl LOAD, um ein bestimmtes Programm von der Diskette zu erhalten. Wenn sich schon ein Programm im Speicher des Computers befindet, löscht das Laden eines Programmes das erste aus dem Speicher. Der Befehl lautet:

```
LOAD "KAPITEL1",8
```

Auch wenn Sie das Programm nicht sehen können, es befindet sich defintiv im Speicher. Wenn Sie es sehen wollen, geben Sie LIST ein. Sie haben diesen Befehl schon mehrfach verwendet und betätigen Sie RETURN.

Die Liste der Dateinamen:

Wenn Sie sich ein Programm ansehen wollen, haben aber dessen Namen vergessen, können Sie sich vom Computer eine Liste aller Dateinamen auf der Diskette anzeigen lassen. Diese Liste ist eine Art Inhaltsverzeichnis; sie zeigt Ihnen, welche Programme auf der Diskette abgespeichert sind. Aber seien Sie gewarnt: der Speicher des

Computers wird gelöscht und enthält dann nur noch die Liste der Dateinamen. Mit anderen Worten, lassen Sie sich nie die Liste der Dateinamen anzeigen, bevor Sie nicht ein Programm, das Sie gerade bearbeiten, gesichert haben. Ansonsten wird das nicht gesicherte Programm gelöscht. Um die Liste der Dateinamen zu sehen, geben Sie ein:

```
LOAD "$",8
```

Es ist nicht ganz einfach, zu erklären, warum LOAD "$",8 die Dateinamen in den Speicher holt. Glauben Sie uns trotzdem, daß es klappt. Mit obigem Befehl erhalten Sie also die Liste der Dateinamen anstelle eines Programms. Probieren Sie den Befehl aus, damit Sie sehen, wie er funktioniert.

Zusammenfassung:
SAVE, VERIFY, LOAD und LIST sind sehr wichtige Befehle, die Sie sich merken sollten. Am Ende jedes Kapitels werden Sie aufgefordert, Ihr Programm zu sichern. Das Programm aus Kapitel 2 wird mit SAVE "KAPITEL2",8 abgespeichert; das Programm aus Kapitel 3 mit SAVE "KAPITEL3",8; etc. Jedes Mal, wenn Sie ein Programm sichern, sollten Sie mit VERIFY kontrollieren, ob es sich auf der Diskette befindet.

Am Anfang jedes Kapitels werden Sie angewiesen, das Programm des vorherigen Kapitels zu LADEN. Wenn Sie Hilfe benötigen, schlagen Sie in dem entsprechenden Abschnitt nach, wie die Befehle funktionieren. Gehen Sie besonders sorgfältig bei der Eingabe der Dateinamen vor (z.B. "KAPITEL1"). Wenn Sie etwa SAVE "KAIPTEL1",8 eingegeben haben, können Sie das Programm aus Kapitel 1 nur dann in den Speicher laden, wenn Sie den Namen wieder falsch eingeben (LOAD "KAIPTEL1",8).

Sie müssen vorsichtig mit Ihren Disketten umgehen. Legen Sie sie nicht in die Sonne oder in die Nähe anderer Wärmequellen, biegen, knicken und verkratzen Sie sie nicht, und halten Sie sie entfernt vom Fernseher oder metallischen Objekten.

Von nun an sollten Sie den folgenden Vorgang am Ende jedes Kapitels ausführen:

1. SAVE "KapitelX",8 (X ist die Nummer des aktuellen Kapitels).
2. Gehen Sie zum nächsten Kapitel über.
3. LOAD "KapitelX",8 (X ist die Nummer des vorherigen Kapitels).

4. LISTen Sie das Programm.
5. Lesen Sie das neue Kapitel.

Da Sie Ihr Programm schon abgespeichert haben und das nächste Kapitel "Sichern auf dem Cassettenrecorder" behandelt, gehen Sie über zu Kapitel 2.

Sichern (SAVE) auf einem Cassettenrecorder

Um Ihr Programm dauerhaft auf einer Cassette zu **sichern**, vergewissern Sie sich, daß Sie ein Band benutzen, das gelöscht werden kann (vorzugsweise ein neues). Das Abspeichern (Sichern) Ihres Programms auf Band gleicht dem Aufzeichnen von Musik auf Band. Wenn Sie ein Lied über ein schon vorhandenes aufzeichnen, löscht das neue Lied das alte.

Da Sie sich ein Programm nicht anhören können, um zu sehen, an welcher Stelle es auf dem Band beginnt und endet, benötigen Sie einen Cassettenrecorder, der mit einem Zählwerk ausgestattet sein muß. Dieses befindet sich oben auf dem Recorder und ist leicht an der Bezeichnung **Counter** zu erkennen. Mit Hilfe des Zählwerks können Sie sich merken, welche Bereiche des Bandes ein schon aufgezeichnetes Programm enthalten. Und so funktioniert es:

Zuerst müssen Sie das Band an den Anfang zurückspulen. Dann setzen Sie das Zählwerk auf 000, indem Sie die kleine Taste daneben drücken.

Sie müssen sich den "Dateinamen" und die "Anfangszahl" des Programms aufschreiben, das Sie sichern wollen. Der Dateiname ist ein Titel, den Sie dem Programm geben, damit der Computer es später leicht finden kann, die Anfangszahl ist die Anzeige auf dem Zählwerk, bevor Sie das Programm sichern. 000 ist die Anfangszahl des ersten gespeicherten Programms auf einem Band.

Wenn ein Programm gesichert wird, beginnt das Zählwerk, sich zu drehen. Dabei zeigt es fortlaufend eine höhere Zahl an. Ist das Pogramm abgespeichert, zeigt das Zählwerk die "Endzahl" des auf dem Band gesicherten Programms an. Sie sollten sich diese Zahl auf einem Zettel notieren.

Wenn Sie das nächste Mal ein Programm sichern, spulen Sie wieder das Band zurück und setzen das Zählwerk auf 000. Dann spulen Sie das Band mit der Vorlauftaste ("Fast Forward", ">>") weiter. Sobald das Zählwerk eine höhere Zahl als die Endzahl des zuletzt abgespeicherten Programms anzeigt, drücken Sie STOP. So

können Sie sicher sein, daß Sie die Bereiche des Bandes, auf denen schon Programme gesichert sind, übergehen.

Um das Programm des Kapitels 1 zu sichern, schreiben Sie sich zuerst die folgenden Spaltenüberschriften auf einen Zettel:

DATEINAME DES PROGRAMMS ANFANGSZAHL ENDZAHL

Den Zettel benötigen Sie, um sich die Stellen des Bandes zu notieren, an denen die Programme abgespeichert sind. So können Sie vermeiden, daß ein Programm bei der Sicherung eines weiteren überschrieben wird. Für das Programm dieses Kapitels schreiben Sie KAPITEL 1 unter "Dateiname des Programms", und 000 unter "Anfangszahl". Die "Endzahl" ergibt sich, wenn das Programm abgespeichert ist.

Um das Programm des Kapitels 1 zu sichern, geben Sie ein:

```
SAVE "KAPITEL1",1
```

Sie müssen KAPITEL1 in Anführungszeichen setzen und dürfen keine zusätzlichen Leerstellen innerhalb der Anführungszeichen einfügen. Wenn Sie diesen Befehl eingegeben haben, betätigen Sie die RETURN-Taste. So teilen Sie dem Computer mit, daß Sie die Eingabe des Befehls beendet haben.

> Beachten Sie: Sie müssen RETURN nach jeder Befehlszeile im Programm-Listing und nach jedem Befehl drücken. Auch wenn wir Sie ab und zu an die Betätigung der RETURN-Taste erinnern, müssen Sie meist selbst daran denken.

Auf dem Bildschirm erscheint dann eine Meldung, daß Sie PLAY und RECORD (Wiedergabe und Aufnahme) bedienen müssen. Vergewissern Sie sich, daß Ihr Cassettenrecorder möglichst weit vom Monitor oder Fernseher entfernt steht und drücken Sie daraufhin gleichzeitig PLAY und RECORD an Ihrem Recorder.

Der Computer beginnt, das Programm zu sichern. Währenddessen werden sämtliche Informationen auf dem Bildschirm gelöscht. Wenn der Computer das Programm abgespeichert hat, erscheint der Text wieder auf dem Bildschirm. Betätigen Sie dann die STOP-Taste an Ihrem Cassettenrecorder. Das Zählwerk zeigt nun die ENDZAHL des Programms, die Sie sich auf Ihrem Zettel notieren sollten. Nun kennen Sie den Anfangs und Endpunkt Ihres Programms KAPITEL1. Wenn Sie das nächste Programm sichern, können Sie sicher sein, daß es hinter dem Ende dieses Programms abgespeichert wird.

Normalerweise wird das Band für den Anfang des nächsten Programms noch etwas weiter als bis zum Ende des vorherigen Programms gespult (ca. 10 Ziffern auf dem Zählwerk).

Der Befehl zur Aufzeichnung Ihres Programms auf dem Band lautet SAVE. Diesem Befehl folgt immer der Dateiname des Programms (Titel) in **Anführungszeichen.** Sie können Ihr Programm mit jedem gewünschten Namen bezeichnen, vorausgesetzt, es überschreitet nicht eine Länge von höchstens 16 Zeichen. Hinter dem Dateinamen geben Sie ein Komma und die Ziffer 1 ein, die dem Computer mitteilt, daß Sie einen Cassettenrecorder benutzen (und nicht ein Diskettenlaufwerk).

Sichern Sie **nie** ein überarbeitetes Programm an Stelle der ursprünglichen Version, es sei denn, das Original ist das zuletzt auf dem Band abgespeicherte Programm. Ein überarbeitetes Programm hat normalerweise eine andere Länge als das Original. Wenn es länger ist, überspielen Sie möglicherweise einen Teil des nächsten Programmes auf dem Band. Wenn es kürzer ist, wird ein Teil des ursprünglichen Programms nie überspielt und wird an das Ende des überarbeiteten Programms angehängt.

Sichern Sie ein überarbeitetes Programm direkt hinter dem letzten Programm auf dem Band. Auf einem Band können Sie denselben Dateinamen so oft Sie wollen benutzen, deshalb kann das überarbeitete Programm denselben Dateinamen wie das Original haben. Sie sollten aber auf Ihrem Zettel notieren, welche die überarbeitete Version ist.

Überprüfung des Abspeichervorgangs:

Wenn Sie ein Programm sichern, sollten Sie kontrollieren, ob es wirklich abgespeichert ist. Dazu benutzen Sie einen weiteren Befehl: VERIFY. Dieser Befehl prüft, ob das Programm sicher auf dem Band abgespeichert ist. Es ist wichtig, daß Sie das Band bis zu einem Punkt vor dem Programmanfang zurückspulen. Spulen Sie bis an den Bandanfang zurück, setzen Sie das Zählwerk auf 000 und spulen Sie (falls nötig) mit der Vorlauftaste bis zu einer Stelle kurz vor dem Anfang des entsprechenden Programms. Wenn das Programm beispielsweise bei 050 beginnt, spulen Sie bis 045. Wenn das Programm bei 000 beginnt (wie das Programm KAPITEL1), spulen Sie **nicht** mit dem Vorlauf weiter.

Um das Programm aus Kapitel 1 zu kontrollieren, spulen Sie das Band zurück, stellen Sie das Zählwerk auf 000 und geben Sie ein:

VERIFY "KAPITEL1",1

Dieses Mal erhalten Sie die Meldung auf dem Bildschirm, PLAY zu betätigen. Folgen Sie diesem Befehl und der Bildschirm wird erneut gelöscht. Einen Augenblick später antwortet der Computer mit FOUND KAPITEL1. Auf diese Weise wissen Sie, daß er das Programm, das Sie überprüfen wollen, gefunden hat. Betätigen Sie nun die C = Taste, um die Kontrolle fortzuführen. Wenn der Computer die Kontrolle beendet hat, erscheinen Cursor und Text wieder auf dem Bildschirm. Betätigen Sie STOP an Ihrem Cassettenrecorder.

Wenn der Computer Ihr Programm nicht finden kann, prüfen Sie, ob Sie das Band weit genug zurückgespult haben. Geben Sie den Befehl VERIFY noch einmal ein. Wenn Sie immer noch nicht die richtige Meldung erhalten, befindet sich das Programm nicht auf dem Band. Listen Sie Ihr Programm, um zu sehen, ob es sich noch im Speicher des Computers befindet (wenn nicht, müssen Sie es neu eingeben), und führen Sie den Abspeichervorgang mit SAVE noch einmal durch. Wenn der Computer Ihr Programm nicht finden kann, haben Sie wahrscheinlich das Band nicht weit genug zurückgespult oder den Befehl SAVE oder VERIFY mit einem Schreibfehler eingegeben.

Laden eines Programmes vom Band:

Da Sie das Programm nun auf dem Band gesichert haben, können Sie eine Pause machen und den Computer ausschalten. Das Programm wird zwar jetzt aus dem Computerspeicher gelöscht, aber es bleibt auf dem Band abgespeichert.

Wenn Sie den Computer wieder einschalten und das Programm in den Speicher zurückholen wollen, laden Sie es vom Band. Beim Laden eines Programmes wird dies nicht vom Band entfernt und im Speicher plaziert, sondern nur in den Speicher kopiert, während das Original des aufgezeichneten Programms auf dem Band erhalten bleibt.

Das Band muß auch hier bis vor den Programmanfang zurückgebracht werden. Spulen Sie das Band an den Anfang zurück, setzen Sie das Zählwerk auf 000 und spulen Sie mit der Vorlauftaste zu einer Stelle kurz vor den Anfang des gewünschten Programms. Beginnt das Programm bei 000, spulen Sie **nicht** mit der Vorlauftaste weiter.

Der Befehl, mit dem Sie ein bestimmtes Programm vom Band erhalten, lautet LOAD. Wenn sich schon ein Programm im Speicher des Computers befindet, wird dieses durch das Laden eines anderen Programms gelöscht. Für den Einsatz des Befehls LOAD spulen Sie das Band zurück, stellen Sie das Zählwerk auf 000 und geben Sie ein:

LOAD "KAPITEL1",1

Der Befehl LOAD ähnelt dem Befehl VERIFY. Wenn die Meldung auf dem Bildschirm erscheint, betätigen Sie PLAY an Ihrem Recorder. Nachdem das zu ladende Programm gefunden ist, betätigen Sie C =, damit der Computer mit dem Laden des Programms fortfahren kann.

Auch wenn Sie das Programm nicht sehen können, es befindet sich defintiv im Speicher. Falls Sie es sehen wollen, geben Sie LIST ein und betätigen Sie RETURN.

Vergewissern Sie sich, daß Sie jedes auf Ihrem Band abgespeicherte Programm notiert haben und bewahren Sie diesen Zettel sicher auf. Der Computer bietet keine Möglichkeit, die Dateinamen der auf einem Band gesicherten Programme anzuzeigen. Kennzeichnen Sie Ihre Bänder und die Listen, damit Sie immer wissen, welche Liste zu welchem Band gehört.

Zusammenfassung:

SAVE, VERIFY, LOAD und LIST sind wichtige Befehle, die Sie sich merken müssen. Am Ende jedes Kapitels sichern Sie Ihr Programm. Das Programm aus Kapitel 2 muß auf dem Band **hinter** dem aus Kapitel 1 abgespeichert werden. Sie sollten sich zu jedem Programm den Dateinamen und die Anfangs und Endzahl, die das Zählwerk anzeigt, aufschreiben. Um das Programm aus Kapitel 2 zu sichern, geben Sie ein: SAVE "KAPITEL2",1; für das Programm aus Kapitel 3: SAVE "KAPITEL3",1; etc. Jedes Abspeichern eines Programms sollten Sie mit VERIFY überprüfen. Zu diesem Zweck muß das Band bis zu einer Stelle kurz vor dem entsprechenden Programmbeginn zurückgespult sein. Dies ist ebenfalls notwendig, um ein Programm zu laden.

Von nun an sollten Sie den folgenden Vorgang am Ende jedes Kapitels ausführen:

1. Notieren Sie sich die Zahl auf dem Zählwerk unter ANFANGS-ZAHL.
2. Geben Sie SAVE "KAPITELX",1 ein (ersetzen Sie X durch die Nummer des Kapitels, das Sie gerade bearbeiten).
3. Notieren Sie sich den DATEINAMEN DES PROGRAMMS (den Dateinamen, den Sie als Kapitel X in Schritt 2 eingegeben haben).
4. Notieren Sie sich die Zahl auf dem Zählwerk, nachdem das Programm gespeichert worden ist.
5. Gehen Sie über zum nächsten Kapitel.

6. Spulen Sie das Band zurück bis zur Anfangszahl des Programms aus dem vorherigen Abschnitt.
7. Laden Sie mit dem Befehl LOAD "KapitelX",1 das Programm (ersetzen Sie X durch die Nummer des vorherigen Kapitels).
8. LISTen Sie das Programm.
9. Lesen Sie das neue Kapitel.

Wenn Sie die Funktion der Befehle einmal vergessen haben sollten, lesen Sie sich dieses Kapitel nochmals durch.

Kapitel 2
Der erste Schritt zur Grafik

Dieses Kapitel ist der erste Schritt zum Nachzeichnen des Bildes auf der Rückseite des Buches. Sie verwenden das in Kapitel 1 eingegebene Programm, aber in diesem Kapitel werden Sie die Befehle PRINT durch Grafik-Unterroutinen ersetzen. Darauf schaltet die erste Unterroutine auf hochauflösende Grafik um, die zweite Unterroutine schaltet sie wieder aus und mit der dritten Unterroutine können Sie die in der hochauflösenden Grafik angezeigten Farben verändern.

Diese drei Unterroutinen stellen die ersten Werkzeuge in Ihrem Grafik-"Werkzeugkasten" dar. Jede von ihnen erledigt eine Aufgabe, die für das Zeichnen eines Bildes hilfreich ist. Nachdem Sie die neuen Unterroutinen eingegeben und geprüft haben, ob sie richtig funktionieren, behandeln wir die neuen Möglichkeiten, die sie eröffnen. Zum Schluß fügen wir unserem Werkzeugkasten ein weiteres Instrument hinzu. Dieses Werkzeug, genannt ZAP-Routine, löscht die HAUPTROUTINE in Ihrem Programm und ermöglicht damit das Eingeben einer neuen Hauptroutine, mit der ein anderes Bild gezeichnet werden kann. Die Unterroutinen bleiben erhalten, um sie beim Zeichnen des neuen Bildes weiter benutzen zu können. Zur Verdeutlichung werden Sie mit ZAP die Hauptroutine Ihres Programms löschen und durch eine neue ersetzen: eine Hauptroutine, die einen Leuchtturm zeichnet.

In diesem Kapitel konzentrieren wir uns auf den Speicher des Computers. Jedes Mal, wenn Sie ein Bild zeichnen, werden die Muster (Punkte und Linien) und Farben des Bildes im Speicher des Computers festgehalten. Sie lernen, wie Ihre Muster und Farben an bestimmten Stellen des Speichers aufbewahrt werden. Dies ist ein wichtiger Bestandteil der Bildherstellung auf dem **Commodore 64.** Der nächste Abschnitt gibt eine kurze Einführung in die Arbeitsweise des Speichers. Wenn Ihnen die Benutzung des Speichers schon vertraut ist, können Sie diesen Abschnitt überschlagen.

Einführung in den Speicher

Der Speicher des Computers besteht aus vielen kleinen "Zellen", deren Aufgabe es ist, Informationen, die alle eine eigene, nur dafür bestimmte Nummer besitzen, aufzunehmen. Um dies besser zu verstehen, stellen Sie sich den Speicher als eine Vielzahl von Briefkästen in einer Straße vor. Jeder Briefkasten kann Informationen (Briefe) aufnehmen und jeder hat seine eigene Nummer (Hausnummer). Insgesamt hat Ihr **Commodore 64** über 64.000 "Briefkästen" zur Speicherung von Informationen. Jeweils 1.000 Briefkästen heißen "1K" Speicher. Also hat Ihr **Commodore 64** einen Speicher von 64K.

Ein Briefträger beginnt seine Route jeden Tag an derselben Stelle und wirft nacheinander Briefe in jeden Briefkasten. Wenn an eine Adresse kein Brief geschickt wurde, läßt der Briefträger diesen Briefkasten aus und geht weiter zum nächsten. Der Computer benutzt seinen Speicher in derselben Weise. Sobald Sie Ihren Computer einschalten und etwas eingeben, zeigt der Computer Ihre Eingabe nicht nur auf dem Bildschirm an, sondern er plaziert die eingegebenen Buchstaben und Symbole auch in einer speziellen Sammlung von Speicherzellen. Auf diese Weise kann er sie später wiederfinden, wenn Sie das Programm starten oder listen. Der Computer beginnt mit der Speicherung des eingegeben Textes immer an derselben Zelle. Während der Eingabe plaziert er die Buchstaben und Symbole in jede Zelle, zu der er gelangt. Geben Sie eine oder mehrere Leerstellen auf dem Bildschirm ein, läßt der Computer eine oder mehrere Speicherzellen leer.

Wenn der Briefträger seine Route beendet hat, werden die Briefe von Ihren Besitzern aus den Briefkästen genommen und am nächsten Tag beginnt der Briefträger seine Route erneut. In dieser Weise arbeitet der Computer, wenn Sie ihn aus- und wieder einschalten. Wenn er ausgeschaltet wird, werden die eingegebenen Buchstaben aus den Speicherzellen herausgenommen (gelöscht). (Deshalb ist es so wichtig, daß Sie Ihre Programme auf Diskette oder Band sichern.) Wenn der Computer wieder eingeschaltet wird, beginnt er bei derselben Anfangszelle und legt die neuen Buchstaben, Symbole und Leerstellen in jeder Zelle ab, zu der er gelangt.

Nun kann der Briefträger sowohl Briefe in die Briefkästen hineinlegen, als auch Briefe herausnehmen, um sie an anderer Stelle abzuliefern. Der Computer verfährt analog dazu. Wenn Sie Ihr Programm eingeben, legt der Computer jedes eingegebene Zeichen in eine seiner Speicherzellen. Benutzen Sie den Befehl LIST, geht der Computer zu

jeder Speicherzelle, sieht nach, welches Zeichen sich dort befindet und zeigt es auf dem Bildschirm an. Wenn er den Inhalt aller Zellen angezeigt hat (er führt dies ziemlich schnell aus), sehen Sie das Programm-Listing auf dem Bildschirm.

Für die Eingabe Ihrer Programmzeilen (Text) ist diese Methode hervorragend geeignet. Wenn Sie aber ein Programm starten, das ein Bild zeichnet, gibt das Programm dem Computer den Grafik-Code, der für die Erstellung des Bildes notwendig ist. Der Computer legt die Grafik-Codes ebenfalls in seinem Speicher ab. Diese Codes müssen in anderen als den Speicherzellen für Ihre Textzeilen abgelegt werden. Wie wird dies bewältigt? Ganz einfach. Indem die "Adressen" der Speicherzellen des Computers verwendet werden, können Sie dem Computer über das Programm mitteilen, wo er Ihr Bild zu speichern und wieder abzuholen hat.

Mit dem Programm in diesem Kapitel führen Sie den Computer zu einer Sammlung von Adressen der Speicherzellen, in denen Sie Ihre Bildmuster und -farben speichern und in denen der Computer sie wiederfindet. Wenn Sie wieder normalen Text eingeben, führen Sie den Computer zurück zu der ursprünglichen Sammlung von Speicherzellen, in denen der Programmtext gespeichert wird.

Es ist wichtig daran zu denken, daß der Computer mehr als eine Sammlung von Speicherzellen gleichzeitig ansehen kann. Wenn Sie also ein Programm starten, das ein Bild zeichnet, sieht der Computer gleichzeitig in die Zellen mit den Programmanweisungen und in die mit den Farbcodes und Bildmustern.

Zuletzt müssen Sie noch wissen, daß einige Zellen bestimmte Funktionen haben und vom Computer fortwährend benutzt werden. Es gibt beispielsweise eine Zelle, in die der Computer immer sieht, sobald sie ihn einschalten. Diese Speicherzelle beinhaltet einen Code, der besagt "alles als normalen Text anzeigen". Wenn Sie also mit der Eingabe beginnen, wird jeder Tastendruck als Text und nicht als Grafikfarbe angezeigt. Diese Zelle kann geändert werden (und wird in diesem Kapitel geändert), so daß der Code dem Computer mitteilt. Grafikfarben an Stelle von Text auf dem Bildschirm anzuzeigen.

Diese und andere Stellen mit Speicherzellen werden in diesem Kapitel ausführlich behandelt. Sie werden die Adressnummer einiger Speicherzellen benutzen, um mit dem Zeichnen Ihres Bildes zu beginnen. Stellen Sie sich den Speicher einfach als eine große Menge von Briefkästen vor, die ihre eigenen Hausnummern haben. Lassen Sie sich nicht verunsichern, wenn der Speicher des Computers immer noch ein vager, abstrakter Begriff für Sie ist. Sie fühlen sich sicherer, wenn Sie beginnen, die Adressen der Speicherzellen zu benutzen, um dem Computer Anweisungen zu geben.

Eingabe der Unterroutinen

Sie benötigen zuerst einmal das Listing des Programms aus Kapitel 1 auf Ihrem Bildschirm. Geben Sie LIST ein und betätigen Sie RETURN. Wenn Sie kein Listing erhalten, benutzen Sie den Befehl LOAD, wie in Kapitel 1 beschrieben, um das Programm in den Speicher zu holen.

Die neuen Programmzeilen, die Sie eingeben müssen, sehen Sie unten. Einige sind völlig neu, andere ersetzen schon im Programm des Kapitels 1 vorhandene Zeilen. Geben Sie die Zeilen genauso ein, wie Sie sie unten sehen. (Benutzen Sie einen Zettel, um die Zeilen zu notieren, die Sie nicht schreiben, wenn Sie diese neuen Zeilen eingeben.) Ändern oder löschen Sie keine der anderen Zeilen im Programm aus Kapitel 1. Falls Sie Schreibfehler korrigieren müssen, lesen Sie in Kapitel 1 den entsprechenden Abschnitt.

```
21 POKE 53265, 59
22 POKE 53272, 29
23 POKE 56576, 198
31 POKE 53265, 27
32 POKE 53272, 21
33 POKE 56576, 199
41 FOR I = 17408 TO 18407
42 POKE I,C
43 NEXT
1005 C = 14
```

Wenn Sie diese neuen Zeilen eingegeben haben, überprüfen Sie das Listing des Programms. Die neuen Zeilen werden automatisch in das Programm eingefügt. Ihr Programm ist aber nun zu lang, um es komplett auf dem Bildschirm zu sehen. Es gibt aber die Möglichkeit Teile des Programm-Listings auf dem Bildschirm zu sehen. Geben Sie LIST 1-44 ein und betätigen Sie RETURN. Sie sehen die Zeilen 1 bis 44 auf Ihrem Bildschirm. Vergleichen Sie sie mit den abgebildeten Programmzeilen. Häufig treten folgende Fehler auf: Eingabe der falschen Zahl mit POKE; Eingabe von O anstelle von 0; Eingabe eines kleinen L (l) anstelle von 1; Eingabe falscher Zeilennummern oder sogar Löschen einer Zeile, die im Programm bleiben muß; Auslassen einer ganzen Zeile. Geben Sie jede fehlende oder falsche Zeile neu ein.

Um den Rest des Programms zu überprüfen, geben Sie LIST 1000 - RETURN ein.

```
1 GOTO 1000
20 REM::::::::::GRAPHICS
21 POKE 53265, 59
22 POKE 53272, 29
23 POKE 56576, 198
24 RETURN
30 REM::::::::::TEXT
31 POKE 53265, 27
32 POKE 53272, 21
33 POKE 56576, 199
34 RETURN
40 REM:::::::::COLORS
41 FOR I = 17408 TO 18407
42 POKE I,C
43 NEXT
44 RETURN
1000 REM::::::::::::::::::
1001 REM       MAIN ROUTINE
1002 REM::::::::::::::::::
1100 GET A$
1110 IF A$ = "G" THEN GOSUB 20
1120 IF A$ = "T" THEN GOSUB 30
1130 IF A$ = "C" THEN GOSUB 40
1140 GOTO 1100
```

Wenn Sie nur einen Teil des Programm-Listings sehen wollen, geben Sie ein:

1. die erste Zeilennummer, die Sie sehen wollen (z.B. 1)
2. einen Bindestrich (-)
3. die letzte Zeilennummer, die Sie sehen wollen (z.B. 44)

Wenn Sie alle Zeilen von einem bestimmten Punkt an sehen wollen, geben Sie ein:

a) die erste Zeilennummer, die Sie sehen wollen (z.B. 1000)
b) nur einen Bindestrich (-)

In beiden Fällen müssen Sie natürlich nach der Eingabe RETURN betätigen.

Wenn Sie glauben, daß Sie das Programm korrekt eingegeben haben, starten Sie es mit RUN. In den folgenden Abschnitten erläutern wir, was auf Ihrem Bildschirm passiert. Wenn Ihr Programm nicht so arbeitet, wie es sollte, betätigen Sie gleichzeitig die RUN/STOP- und die RESTORE-Taste. LISTen Sie Ihr Programm und überprüfen Sie jede Zeile noch einmal.

Als erstes entfernt der Computer den Cursor vom Bildschirm (wenn nicht, überprüfen Sie Zeile 1 und Zeilen 1000-1140 Ihres

Programms). Wie im vorherigen Kapitel wartet der Computer auf die Betätigung einer bestimmten Taste für GET A$. Diese Zeile haben Sie in Ihrem Programm nicht geändert. Die Betätigung von G,T und C erzeugt auch hier eine sichtbare Antwort des Computers, die sich aber von der in Kapitel 1 unterscheidet, da Sie die Unterroutinen geändert haben. Folgendes passiert:

Betätigen Sie G. Der Computer schaltet automatisch auf hochauflösende Grafik um. Es hängt von der Version Ihres **Commodore 64** ab, was der Bildschirm anzeigt. Wenn Ihr Bildschirm Farben anzeigt, entweder in einer Art Muster oder nur als zackige, ungeordnete Linien, ist alles in Ordung. (Wenn Sie keine Farben auf dem Bildschirm sehen, überprüfen Sie die Zeilen 1110 und 20-24 in Ihrem Programm.) Gehen Sie ganz dicht an Ihren Bildschirm heran. Sie müßten in der Lage sein, die winzigen Pixels zu sehen, die wir schon erwähnt haben.

Betätigen Sie C. Ihr Bildschirm ändert Zeile für Zeile die Farben. Sie sehen eine Mischung aus Blau und Schwarz. Die blaue Farbe bildet die "Hintergrund"-Farbe des Bildschirms, die schwarze die "Vordergrund"- Farbe. In diesem Kapitel werden Sie eine Möglichkeit erlernen, die Hinter- und Vordergrundfarben zu ändern. In späteren Kapiteln lernen Sie, zu bestimmen, welche Pixels die Hintergrund- und welche die Vordergrundfarbe bilden. (Wenn C die Farben auf Ihrem Bildschirm nicht ändert, überprüfen Sie die Zeilen 1130 und 40-44 in Ihrem Programm.)

Betätigen Sie noch einmal C. Was geschieht? Hoffentlich nichts. Das neue Programm läßt den Computer die Farben des hochauflösenden Bildschirms in Blau und Schwarz ändern, sobald C betätigt wird. Sie haben mit dieser Taste die Farben auf dem Bildschirm schon in Blau und Schwarz geändert, deshalb darf die erneute Betätigung der Taste nichts auf dem Bildschirm bewirken.

Betätigen Sie T. Die hochauflösende Grafik wird "ausgeschaltet" und der Computer kehrt zur normalen Textanzeige zurück. (Wenn nicht, überprüfen Sie die Zeilen 1120 und 30-34 in Ihrem Programm.) Sie können aber noch nicht LIST oder einen anderen Befehl eingeben, da das Programm weiterläuft. Betätigen Sie zur Überprüfung G.

Probieren Sie dieses Programm aus, solange Sie wollen. Denken Sie daran, G schaltet die hochauflösende Grafik ein, T schaltet sie wieder aus und Sie erhalten Textanzeige und C ändert die Farben des hochauflösenden Bildschirms in Blau und Schwarz (das dauert eine Weile und wird unabhängig davon durchgeführt, ob der Bildschirm bereits blau und schwarz ist). Nachdem Sie nun gelernt haben, wie das Programm arbeitet, werden Sie es anhalten wollen. Wenn Sie T betätigt und wieder die Textanzeige erhalten haben, drücken Sie, wie gewöhnlich, RUN/STOP.

Beachten Sie: Wenn Sie den Grafik-Bildschirm sehen, betätigen Sie gleichzeitig RUN/STOP und RESTORE. Dies ist der einzige Weg, **direkt** vom hochauflösenden Bildschirm zurück zu Ihrem Programm zu gelangen. Benutzen Sie die RUN/STOP RESTORE-Kombination als "Nottaste", wenn Sie Probleme haben, zum Textbildschirm zurückzugelangen.

Aufgliederung der Unterroutinen

Geben Sie LIST 1-44 RETURN und LIST 1005 RETURN ein, um das Listing der neuen Programmzeilen zu erhalten. Der Rest des Programms wird ausgelassen. Es handelt sich um die Hauptroutine, die ein GOSUB 20 ausführt, wenn G betätigt wird, ein GOSUB 30 bei T und ein GOSUB 40 bei C.

Der Computer benötigt zwei Informationen, um Text oder Grafik anzuzeigen. Für die Anzeige von Textzeilen muß er wissen:

1. wo er die Textzeichen findet,
2. wo er die Farben dieser Textzeichen findet.

Für die Anzeige von Grafik muß er wissen:

1. wo er Ihr Bildmuster findet,
2. wo er die Farben für Ihr Bildmuster findet.

Sie müssen also jedes Mal, wenn der Computer Text oder Grafik anzeigen soll, angeben, wo diese Informationen zu finden sind. An dieser Stelle nehmen die Unterroutinen ihre Tätigkeit auf. Wir erklären nun **wie** diese Unterroutinen arbeiten, während das **warum** ihrer Funktionsweise in drei abgeteilten Abschnitten jeweils für eine Unterroutine erklärt wird. Wenn Sie an einer technischen Erklärung dieser Werkzeuge interessiert sind, sollten Sie sich die jeweiligen Abschnitte durchlesen.

Jede der Unterroutinen enthält eine Reihe von POKE-Befehlen. POKE teilt dem Computer mit, zu einer bestimmten Speicherzelle zu gehen und einen Code darin zu plazieren. Das ist alles. Auf POKE folgen immer zwei Zahlen, getrennt durch ein Komma. Die erste Zahl ist die **Adresse** der Speicherzelle, die zweite Zahl ist der Code, den die Speicherzelle enthalten soll. Dieser Code teilt dem Computer mit, jedes Mal, wenn er in die Speicherzelle sieht, etwas bestimmtes auszuführen. Dies alles wird gleich noch verständlicher. Merken Sie sich aber, daß jede Adresse und jeder Code einer Speicherzelle, die Sie in Ihren Unterroutinen verwenden, eben solche sind, die der Computer kennt und versteht. Mit anderen Worten, Sie können nicht Ihre eigenen Speicheradressen und -codes aufstellen.

Die erste Unterroutine befindet sich in den Zeilen 20 bis 24. Der Computer führt die Aufgabe der Unterroutine aus, wenn G betätigt wird, während das Programm läuft. Drei Dinge passieren in dieser Unterroutine. Als erstes schaltet der Computer auf hochauflösende Grafik um (Zeile 21). Dies ist ein sehr wichtiger Bestandteil des Werkzeugs und jedes Programms, das ein Bild anzeigen soll.

Die nächsten beiden Zeilen (22 und 23) teilen dem Computer mit, wo er Ihre Bildmuster und -farben speichern soll und wiederfindet. Der Speicher des Computers besteht aus 4 Hauptteilen. Diese Teile werden mit **bank** bezeichnet, wobei jeder Teil 16.000 Speicherzellen enthält. Diese Banks sind mit 0, 1, 2 und 3 numeriert. Bank 0 wird benutzt, um alle Arten von Informationen zu speichern, einschließlich Ihres Programmtextes. Da Bank 0 eine Menge Informationen speichert, befinden sich nicht viele leere Speicherzellen darin. Um möglichst viel Platz für die Speicherung Ihres Bildes zur Verfügung zu haben, sollten Sie es zu Bank 1 leiten, wo viele leere Speicherzellen vorhanden sind.

Die Zeile 22 teilt dem Computer mit, wo er Ihre Bildmuster und -farben in den Banks findet. Der Computer wird angewiesen, egal in welchem Bank er nachsieht, immer an einer bestimmten Stelle in diesem Bank zu beginnen, um Ihre Bildmuster und -farben zu erhalten. Die Zeile 23 lenkt Ihre Farbcodes und Bildmuster zu Bank 1. Da die Zeile 22 dem Computer mitteilt, wo er Ihre Bildmuster und Farbcodes innerhalb eines Banks findet, und die Zeile 23 ihm mitteilt, in welchem Bank sie sich befinden, hat der Computer alle notwendigen Informationen zum Abspeichern und Abrufen Ihres Bildes. Wenn diese Zeilen aus der Unterroutine ausgelassen werden, erhalten Sie am Ende ein verzerrtes Bild. Der Computer hat ohne die Zeilen keine Möglichkeit, herauszufinden, welche Zellen die Bildmuster und welche die Farbcodes enthalten.

Eine technische Erläuterung dieser Unterroutine finden Sie in folgendem Abschnitt. Sie sollten die Abschnitte **Zweck** und **Anwendung** lesen. Solange Sie wissen, daß diese Unterroutine in jedem Programm notwendig ist, um ein Bild auf Ihrem Bildschirm anzuzeigen (anstelle von Text), brauchen Sie die **Technische Beschreibung** nicht zu lesen.

Werkzeug 20 :::::::::: Grafik

```
20 REM::::::::::GRAPHICS
21 POKE 53265, 59
22 POKE 53272, 29
23 POKE 56576, 198
24 RETURN
```

__Zweck:__ Dieses Werkzeug schaltet auf hochauflösende Grafik um, damit Ihr Bild in Grafikfarben erscheint und nicht Symbole und Zeichen (Text) angezeigt werden.

__Anwendung:__ Um dieses Werkzeug zu benutzen, benötigen Sie nur eine GOSUB 20 Anweisung in Ihrem Programm. Wenn die einzige Funktion Ihres Programms darin besteht, ein Bild zu zeichnen, muß der Befehl GOSUB einer der ersten in Ihrem Programm sein.

__Technische Beschreibung:__ POKE ist ein BASIC-Befehl, der einen Code an einer Stelle im Computer-Speicher plaziert. Die Speicherstelle 53265 bestimmt immer, ob der Computer Grafik oder Text anzeigen soll. Der Code 59, der an dieser Stelle plaziert wird (POKE 53265,59), teilt dem Computer mit, Farbgrafik anzuzeigen.

Die Speicherstelle 53272 bestimmt immer, wo der Computer die Pixelmuster und -farben innerhalb eines Banks speichert und findet. Eine 29 an der Speicherstelle 53272 (POKE 53272,29) teilt dem Computer die richtige Stelle zur Speicherung der Farbcodes und Pixelmuster innerhalb des Banks mit. Der Computer weiß also auch später, wo er sie wiederfindet. __Vorsicht:__ Andere Zahlen als 29 können bewirken, daß die Farbcodes mitten in Ihren Pixelmustern gespeichert werden. Dies würde Ihr Bild zerstören, da der Computer versuchen würde, die Farbcodes auch als Pixelmuster zu benutzen.

Die Speicherstelle 56576 bestimmt, welches der Banks der Computer benutzen soll, um Ihre Grafikcodes zu speichern und abzurufen. Beim Commodore 64 stehen 4 Banks zur Verfügung, jedes mit 16K Speicherkapazität. Die 198 an der Speicherstelle 56576 (POKE 56576,198) teilt dem Computer mit, Ihr Bild in Bank 1 zu speichern und es dort abzurufen. Normalerweise würde das Bild zu Bank 0 gelenkt, die jedoch zur Speicherung Ihrer Programmzeilen benutzt wird und deshalb wenig ungenutzten Platz zur Verfügung hat.

Mit der nächsten Unterroutine (Zeilen 30 bis 34) kehren Sie zur Textanzeige zurück. Dazu wird alles aus der Grafik-Unterroutine widerrufen. Beachten Sie, daß die Speicherstellen in dieser Unterroutine (53265, 53272 und 56576) genau dieselben sind, wie in der Grafik-Unterroutine. Unterschiedlich ist aber der Code, den der Computer an diese Speicherstellen setzen soll.

Mit der nächsten Unterroutine (Zeilen 30 bis 34) kehren Sie zur Textanzeige zurück. Dazu wird alles aus der Grafik-Unterroutine widerrufen. Beachten Sie, daß die Speicherstellen in dieser Unterroutine (53265, 53272 und 56576) genau dieselben sind, wie in der Gra-

fik-Unterroutine. Unterschiedlich ist aber der Code, den der Computer an diese Speicherstellen setzen soll.

Werkzeug 30 :::::::::: Text

```
30 REM::::::::::TEXT
31 POKE 53265, 27
32 POKE 53272, 21
33 POKE 56576, 199
34 RETURN
```

Zweck: *Dieses Werkzeug schaltet die hochauflösende Grafik aus, und Sie erhalten wieder Textanzeige.*

Anwendung: *Um dieses Werkzeug zu benutzen, benötigen Sie den Befehl GOSUB 30 in Ihrem Programm. Wenn die einzige Funktion Ihres Programms darin besteht, ein Bild zu zeichnen, muß der Befehl GOSUB einer der letzten in Ihrem Programm sein. Wie Sie später noch sehen werden, wird die Hauptroutine geändert, so daß diese Unterroutine nur ausgeführt werden kann, wenn die LEERTASTE betätigt wird.*

Technische Beschreibung: *POKE ist ein BASIC-Befehl, der einen Code an einer Stelle im Computer-Speicher plaziert. Die Speicherstelle 53265 bestimmt, ob der Computer Grafik oder Text anzeigen soll. Der Code, der an dieser Stelle plaziert wird (POKE 53265,27) teilt dem Computer mit, Text anzuzeigen.*

Die Speicherstelle 53272 bestimmt immer, wo der Computer die Buchstaben, Symbole und Farben für die Anzeige des Textes speichert und findet. Der Computer erwartet, daß sich Textinformation immer an einer Stelle befindet. Wenn Sie versuchen, ihm mitzuteilen, daß Ihre Textinformation sich an anderer Stelle befindet, versteht er dies nicht. Deshalb **müssen** *Sie den entsprechenden Code an der Speicherstelle 53272 ersetzen, damit wieder Text angezeigt wird. Der Code lautet 21 (POKE 53272,21).*

Die Speicherstelle 56576 bestimmt, welches der Banks der Computer benutzen soll, um Ihren Text abzuspeichern und abzurufen. Beim Commodore 64 stehen 4 Banks zur Verfügung, jedes mit einer Speicherkapazität von 16K. Die 199 an der Speicherstelle 56576 (POKE 56576,199) teilt dem Computer mit, Text in Bank 0 zu speichern und ihn dort abzurufen.

Zum Schluß kommen wir zur Unterroutine in den Zeilen 40-44. Diese Unterroutine wird benutzt, um die Vorder- und Hintergrundfarben des gesamten Grafikbildschirmes zu ändern.

Ihre Farbcodes befinden sich jetzt in 1.000 verschiedenen Speicherzellen. Jede dieser Speicherzellen hat eine bestimmte Adresse, von 17408 bis 18407. Jede Speicherzelle steuert einen Block mit winzigen Lichtpunkten. Wenn Sie den Farbcode in Speicherzelle 17408 ändern, ändert der erste Block mit Lichtpunkten seine Farbe. Ändern Sie den Farbcode in Speicherzelle 17409, ändert der zweite Block seine Farbe. Wenn Sie die Farbe aller Blöcke mit Lichtpunkten ändern wollen, müssen Sie in jede Speicherzelle von 17408 bis 18407 einen neuen Farbcode eingeben. Genau dies führt die Unterroutine aus.

Die Zeile 41 lautet: FOR I = 17408 TO 18407. Mit dieser Zeile wird dem Buchstaben "I" jedesmal, wenn der Computer an diese Zeile gelangt, eine Zahl zugewiesen. Das erste Mal wird I die Zahl 17408 zugewiesen, das zweite Mal 17409, das dritte Mal 17410, etc., bis I zum Schluß die Zahl 18407 zugewiesen wird. Wie Sie sehen, sind diese Zahlen die Speicherstellen, die Ihre Grafik-Farbcodes speichern.

Mit dem Befehl POKE können Sie den Farbcode an jeder dieser Speicherstellen ändern. Die auf dem Bildschirm angezeigten Farben werden mit dem C in POKE I,C bestimmt. Wenn Sie die Zeile 1005 anschauen, sehen Sie, daß C = 14 ist. 14 ist der Farbcode, der als Hintergrundfarbe Blau und als Vordergrundfarbe Schwarz angibt. In Zeile 43 sehen Sie das Wort NEXT. Dies teilt dem Computer mit, die nächste Zahl für I zu nehmen und wieder die Unterroutine zu durchlaufen. Auf diese Weise können sehr schnell 1000 POKE Befehle geschrieben werden (POKE 17408,14; POKE 17409,14; POKE 17410,14; etc.).

Wie die Vordergrund/Hintergrund-Farben bestimmt werden, können Sie der Farbtabelle für hochauflösende Grafik im Anhang dieses Buches entnehmen. Die über jeder Spalte genannten Farben bestimmen die Hintergrundfarbe des Bildschirms, die auf der linken Seite genannten Farben die Vordergrundfarbe. Die Zahlen an den Schnittpunkten jeder Vorder- und Hintergrundfarbe können benutzt werden, um die Farbe des Bildschirmes zu ändern. Suchen Sie die Nummer 14 auf der Karte. Sie befindet sich ganz rechts in der ersten Reihe. Direkt über der 14 sehen Sie die Hintergrundfarbe **Blau2.** Die Vordergrundfarbe in dieser Zeile ist **Schwarz.** Diese sind die auf dem Bildschirm angezeigten Farben.

Um das Programm für eine andere Farbenkombination zu ändern, identifizieren Sie C mit einer anderen Zahl aus der Farbkarte.

Farbkarte für die hochauflösende Grafik

Hintergrundfarben

Vordergrundfarben	Schwarz	Weiß	Rot	Cyan	Purpur	Grün	Blau	Gelb	Orange	Braun	Rot 2	Grau 1	Grau 2	Grün 2	Blau 2	Grau 3
Schwarz	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Weiß	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Rot	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
Cyan	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
Purpur	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
Grün	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
Blau	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
Gelb	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
Orange	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
Braun	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
Rot 2	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
Grau 1	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
Grau 2	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
Grün 2	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
Blau 2	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
Grau 3	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255

Angenommen, Sie wünschen Rot als Hintergrund- und Gelb als Vordergrundfarbe. Welche Zahl müssen Sie mit C gleichsetzen? Wenn Sie 114 herausgesucht haben, liegen Sie richtig. Ändern Sie die Zeile 1005 in C = 114 und starten das Programm erneut mit RUN. Nachdem Sie die unvermeidliche G-Taste gedrückt haben, um die hochauflösende Grafik einzuschalten, betätigen Sie C für den Wechsel der Farben.

Da Sie nun wissen, wie die Farben auf dem Bildschirm geändert werden, wünschen wir Ihnen viel Spaß beim Experimentieren. Vergewissern Sie sich, daß Sie nachher die Zeile 1005 wieder in 1005 C = 14 umändern.

Werkzeug 40 :::::::::: Farben

```
40 REM:::::::::COLORS
41 FOR I = 17408 TO 18407
42 POKE I,C
43 NEXT
44 RETURN
```

Zweck: *Dieses Werkzeug gibt dem Hinter- und Vordergrund auf dem gesamten Bildschirm je eine Farbe, wobei Zeile 1005 (C = 14) die anzuzeigende Farbkombination bestimmt.*

Anwendung: *Wollen Sie dieses Werkzeug benutzen, benötigen Sie den Befehl GOSUB in Ihrer Hauptroutine und eine Zeile, die den Code von C definiert (z.B. 1005 C = 14). C kann mit jeder Kombination von Vorder- und Hintergrundfarben gleichgesetzt werden, die Sie in der Farbkarte in diesem Kapitel finden.*

Technische Beschreibung: *Jede Speicherstelle zwischen 17408 und 18407 bestimmt die Vorder-/Hintergrundfarben eines Blockes von 8 x 8 Pixels auf Ihrem Grafikbildschirm. Diese Blöcke beginnen oben links auf dem Bildschirm und setzen sich von links nach rechts den gesamten Bildschirm hinunter fort. Die Farben jedes Blockes werden durch den Farbcode an jeder der Speicherstellen zwischen 17408 und 18407 bestimmt. Die Speicherstelle 17408 bestimmt die Farbe des Blockes oben links auf dem Bildschirm, die Speicherstelle 17409 bestimmt die Farbe des nächsten Blockes rechts, etc. Der letzte Block auf dem Bildschirm (unten rechts) wird von der Speicherstelle 18407 bestimmt. Die Zeile 41 erhöht I jedes Mal, wenn der Befehl NEXT vom Computer gelesen wird; dh., daß I bei 17408 beginnt, wenn die Zeile 41 das erste Mal gelesen wird. Die POKE Befehle*

plazieren einen entsprechenden Farbcode an der Speicherstelle, die durch den aktuellen Wert von I angegeben wird. C ist der Farbcode, der an jeder Speicherstelle plaziert wird. Durch Änderung des Wertes von C in Zeile 1005 ändern sich die Vorder- und Hintergrundfarben des Grafikbildschirms.

Auswahl der Farben für Ihr Bild

Sie haben gerade gelernt, daß Informationen über die Farben in den Speicherstellen 17408 bis 18407 gespeichert werden. Denken Sie daran, daß diese 1.000 Stellen nichts über das Aussehen des zu zeichnenden Bildes aussagen, sondern allein die Farben bestimmen, die für das Bild benutzt werden.

In jedem Block von 8 x 8 Pixels können Sie zwei Farben verwenden. Die eine färbt den Hintergrund des Bildes, wie z.B. Blau für den Himmel, während die zweite Farbe als eigentliche Zeichenfarbe genutzt wird. Die erste wird mit **Hintergrundfarbe** bezeichnet, die zweite, für das eigentliche Bild, die **Vordergrundfarbe**, da sie benutzt wird, um Formen auf der Hintergrundfarbe zu zeichnen.

Einige Kombinationen von Vorder-/Hintergrundfarben erzeugen unscharfe Bilder auf dem **Commodore 64.** Der Farbcode 37 (Rot und Grün), der in der Farbtafe normal gedruckt ist, steht für ein Farbenpaar, das nicht so gut zusammenpaßt. Der Farbcode 54 (Cyan und Blau), der in der Farbtafel dunkel unterlegt ist, erzeugt hingegen ein scharf abgegrenztes Bild. Es kommt ganz auf das Bild an, das Sie zeichnen wollen, ob Sie scharfe oder verschwommene Umrisse wünschen.

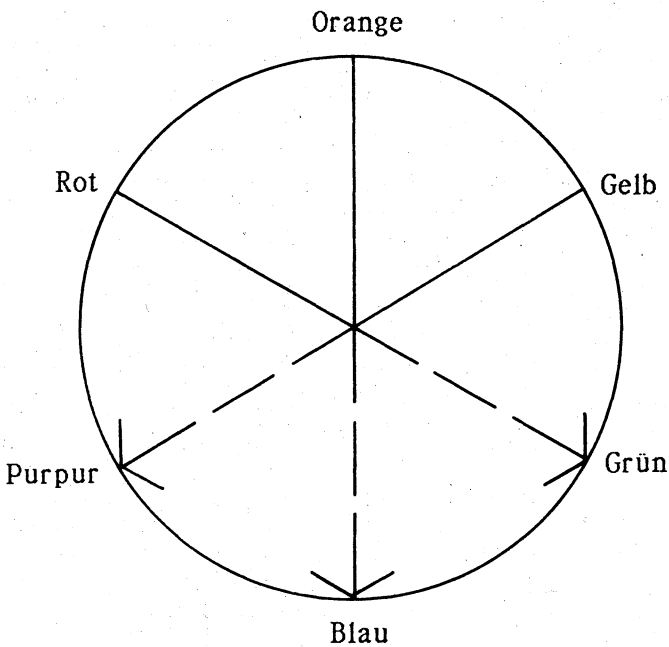
Unser Bild mit dem Schiff im Wasser ist ein ziemlich naturgetreues Bild. Diese Art bildlicher Darstellungen wird als "figürlich" bezeichnet, weil sie als Abbildungen der Wirklichkeit sehr nahe kommen. Da unser Bild möglichst naturgetreu aussehen soll, wählen wir eine Farbkombination, die scharf abgegrenzte Umrisse erzeugt (Schwarz vor Blau).

Darüberhinaus können Farben dazu benutzt werden, den Eindruck von Dreidimensionalität hervorzubringen; d.h. einige Formen scheinen weit entfernt, andere widerum in unmittelbarer Nähe. Wenn Sie für den Hintergrund eine dunkle und für die Formen im Vordergrund eine helle Farbe wählen, erscheint jener im allgemeinen weit entfernt. Denken Sie an ein Raumschiff und Planeten, gezeichnet in hellen Farben vor einem dunklen Hintergrund. Das Raumschiff und die

Planeten scheinen Ihnen näher zu sein, als der dunkle Hintergrund des Weltraums.

Bei einigen Farbkombinationen besteht zudem ein krasser Unterschied, ein sogenannter "Kontrast" zwischen zwei Farben. Kontrastierende Farben sind Schwarz und Weiß, Purpur und Gelb, Rot und Grün, um nur einige zu nennen. Völlig entgegengesetzte, kontrastierende Farben werden als "komplementär" bezeichnet. Die Benutzung komplementärer Farben erzeugt folglich die größten Kontraste. Einige komplementäre Farbenpaare sind:

Warme und kalte Komplementärfarben



Rot und Grün
Orange und Blau
Gelb und Purpur

Einige Farben (wie Rot, Orange und Gelb) werden als "warme" Farben bezeichnet, andere (wie Grün, Blau und Purpur) als "kalte" Farben. Warme Farben werden mit Dingen assoziiert, die auch in Wirklichkeit warm sind - Sonne, ein sonniger Tag, Feuer. Kalte Farben werden für Objekte verwendet, die normalerweise als kühl oder kalt empfunden werden - Eis, kaltes Wasser. Künstler benutzen Farben sehr bewußt, so daß der Betrachter eines Bildes etwas bestimmtes fühlt. So kann die Verwendung heißer, feuriger Farben eine intensive, bewegte Reaktion auf das Bild bewirken, wohingegen kalte Farben in einem Bild entspannende, beruhigende Wirkung zu erzielen vermögen.

Da die Auswahl der Farben für Ihre Bilder oft ebenso wichtig wie die Abbildung selbst ist, überlegen Sie sich die mögliche Wahrnehmung des Bildes durch den Betrachter vorher (wirklichkeitsnah, neblig verschwommen o.ä.). Des weiteren sollten Sie an die gewünschte Wirkung der Abbildung selbst denken: eine ruhige und friedliche Athmosphäre wie ein kühler See im Sommer oder anders, eine heiße und feurige Stimmung wie ein Sonnenuntergang? Wenn Sie die oben beschriebenen Techniken bezüglich der Farben benutzen, können Sie die Gefühle des Betrachters beim Anschauen Ihrer Bilder besser steuern.

Eingabe der ZAP-Unterroutine

Sie werden nun die ZAP-Routine eingeben. Dieses Werkzeug erweist sich dann als nützlich, wenn Sie ein neues Bild zeichnen wollen. Seine einzige Aufgabe besteht darin, Ihre existierende Hauptroutine zu löschen, ohne daß die Unterroutinen angetastet werden.

Bevor Sie mit der Eingabe beginnen, ein Wort der Warnung: Geben Sie diese Routine sehr sorgfältig ein und überprüfen Sie sie - besser einmal mehr als zuwenig. Wenn diese Routine falsch eingegeben und das Programm gestartet wird, kann eine ganze Menge schiefgehen. Seien Sie besonders sorgfältig bei Zeile 11 (A = 256: B = 2049: C = 1003). Vergewissern Sie sich, daß Sie C = 1003 und keine andere Zahl eingeben. Während der Eingabe wird jede Zeile, die für den Bildschirm zu lang ist, automatisch in der nächsten Zeile fortgeführt.

```
10 REM :::::::::ZAP
11 A = 256: B = 2049: C = 1003
12 IF PEEK(B+2) + A * PEEK(B+3) >= C THEN 15
13 B = PEEK(B) + A * PEEK(B+1): ON ABS(B<>0)
   GOTO 12: END
```

```
14 A = 256: B = PEEK(251) + A * PEEK(252)
15 IF PEEK(B+1) = 0 THEN END
16 PRINT CHR$(147) PEEK(B+2) + A * PEEK(B+3):
   PRINT " GOTO 14"
17 POKE 251, B - INT(B/A) * A: POKE 252,B/A
18 POKE 631,19: POKE 632,13: POKE 633,13:
   POKE 198,3: END
```

Nachdem Sie jede Zeile dieser Routine eingegeben und überprüft haben, können Sie sie mit der Eingabe eines gesonderten, nicht für das gesamte Programm geltenden RUN-Befehls ausprobieren. Erinnern Sie sich noch, daß Sie das Listing eines Teils Ihres Programms, beginnend bei Zeile 100 erhalten konnten? Genauso können Sie den Computer auch Teile Ihres Programms ausführen lassen, beginnend bei einer bestimmten Zeilennummer. Der Computer soll die Zeilen 10 bis 18 ausführen. Wenn Sie sich die Zeile 18 ansehen, erkennen Sie, daß sie mit dem Wort END aufhört. Dieses Wort hat dieselbe Wirkung, wie die Betätigung von RUN/STOP.

Geben Sie RUN 10 ein und drücken Sie RETURN. Die 10 teilt dem Computer mit, an welcher Stelle er das Programm zu starten hat. Das END in Zeile 18 stoppt den Computer.

Als erstes wird Ihr Programm vom Bildschirm gelöscht. Oben links sehen Sie mehrere blinkende Begriffe. Zuerst die Zeilennummern, die Sie löschen (Zeilen 1005, 1110, 1120, 1130 und 1140), dann GOTO 14 und READY, zuletzt blinkt der Cursor. Wenn der Computer die Ausführung der Routine beendet hat, sehen Sie auf dem Bildschirm:

```
1140
GOTO 14
READY
```

Geben Sie zuerst LIST ein, darauf RETURN, und Sie sehen das Ergebnis: alle Unterroutinen sind noch vorhanden, die Hauptroutine (ab Zeile 1005) fehlt. Vergleichen Sie mit den unten angeführten Programmzeilen, um sich zu vergewissern, daß die ZAP-Routine keine notwendigen Unterroutinen gelöscht hat.

```
1 GOTO 1000
10 REM :::::::::ZAP
11 A = 256: B = 2049: C = 1003
12 IF PEEK(B+2) + A * PEEK(B+3) >= C THEN 15
13 B = PEEK(B) + A * PEEK(B+1): ON ABS(B<>0)
   GOTO 12: END
14 A = 256: B = PEEK(251) + A * PEEK(252)
```

```
15 IF PEEK(B+1) = 0 THEN END
16 PRINT CHR$(147) PEEK(B+2) + A * PEEK(B+3):
   PRINT " GOTO 14"
17 POKE 251, B - INT(B/A) * A: POKE 252,B/A
18 POKE 631,19: POKE 632,13: POKE 633,13:
   POKE 198,3: END
20 REM::::::::::GRAPHICS
21 POKE 53265, 59
22 POKE 53272, 29
23 POKE 56576, 198
24 RETURN
30 REM::::::::::TEXT
31 POKE 53265, 27
32 POKE 53272, 21
33 POKE 56576, 199
34 RETURN
40 REM::::::::::COLORS
41 FOR I = 17408 TO 18407
42 POKE I,C
43 NEXT
44 RETURN
1000 REM::::::::::::::::::
1001 REM       MAIN ROUTINE
1002 REM::::::::::::::::::
```

Wenn Ihr Computer nicht, wie oben angegeben, antwortet oder einige Ihrer Unterroutinenzeilen nun fehlen, überprüfen Sie jede Zeile der ZAP-Routine. Geben Sie jede Zeile der ZAP-Routine neu ein, die einen Fehler aufweist. Wenn Ihre ZAP-Routine eine oder mehrere andere Unterroutinenzeilen gelöscht hat, geben Sie diese ebenfalls neu ein.

Dieses Werkzeug wird im folgenden, abgetrennten Abschnitt noch ausführlicher behandelt. Wichtig ist, daß Sie zwei Dinge verstehen, wenn Sie dieses Werkzeug benutzen: 1) verkörpert es eine schnelle und einfache Methode, Ihre Werkzeuge zum Zeichnen eines neuen Bildes aufzubewahren, 2) befiehlt die Zeile 1 dem Computer direkt zu Zeile 1000 zu gehen, wenn das Programm gestartet wird. Diese GOTO-Zeile ist jetzt besonders wichtig. Würde sie nämlich ausgelassen, würden die Unterroutinen beim Start des Programms automatisch als erste ausgeführt und Ihre Hauptroutine erst einmal gelöscht, da ja die ZAP-Routine bei Zeile 10 beginnt.

```
10 REM ::::::::::ZAP
11 A = 256: B = 2049: C = 1003
12 IF PEEK(B+2) + A * PEEK(B+3) >= C THEN 15
13 B = PEEK(B) + A * PEEK(B+1): ON ABS(B<>0)
   GOTO 12: END
14 A = 256: B = PEEK(251) + A * PEEK(252)
15 IF PEEK(B+1) = 0 THEN END
16 PRINT CHR$(147) PEEK(B+2) + A * PEEK(B+3):
   PRINT " GOTO 14"
17 POKE 251, B - INT(B/A) * A: POKE 252,B/A
18 POKE 631,19: POKE 632,13: POKE 633,13:
   POKE 198,3: END
```

Zweck: *Diese Routine löscht alle Zeilen Ihrer Hauptroutine, beläßt aber jede Ihrer Unterroutinen in Ihrem Programm, so daß Sie sie für die Zeichnung eines anderen Bildes benutzen können.*

Anwendung: *Um diese Routine anwenden zu können, benötigen Sie eine GOTO Anweisung an einer Stelle vor Anfang der ZAP-Routine. So können Sie Ihr Programm auch starten, ohne daß die ZAP-Routine ausgeführt wird. Wenn Sie die Hauptroutine löschen wollen, geben Sie RUN und die Zeilennummer ein, an der die ZAP-Routine beginnt (z.B. RUN 10 RETURN).*

Technische Beschreibung: *Die ZAP-Routine stellt ein sehr nützliches Werkzeug dar. In diesem Abschnitt erklären wir nur,* **was** *die ZAP-Routine bewirkt.* **Wie** *sie funktioniert, wird in diesem Grafikbuch für der Komplexität der Routine nicht behandelt. Ihr Wert ist aber zu groß, um eine Erklärung völlig auszulassen, deshalb geben wir Ihnen zumindest eine allgemeine Erläuterung ihrer Arbeitsweise.*

Normalerweise müssen Sie jede Zeilennummer eingeben und RETURN betätigen, um eine Reihe von Zeilen aus Ihrem Programm zu löschen. Wenn viele Zeilen zu löschen sind (wie es voraussichtlich in Ihrer Hauptroutine der Fall sein wird), dauert dies eine Weile.

Eine Alternative wäre ein Programm, das jede Zeilennummer auf den Bildschirm gibt, und wartet, daß Sie RETURN betätigen. Noch besser wäre ein Programm, das für Sie RETURN nach jeder Zeilennummer betätigt. Wenn ein Programm so arbeitet, können Sie sich zurücklehnen und zusehen. Nichts anderes ist die ZAP- Routine: ein kleines innerhalb eines größeren Programms. Deshalb wird die Routine mit RUN ausgeführt.

Der Leuchtturm

Wie versprochen zeichnen wir nun vor Beendigung des Kapitels noch etwas auf den Bildschirm. Da Sie Ihre Hauptroutine mit ZAP gelöscht haben, muß eine neue eingegeben werden. Sie finden sie unten angegeben. Bevor Sie mit der Eingabe beginnen, sehen Sie sich die Zeile 5010 an. Sie enthält IF A$ = " " THEN 6000. Achten Sie darauf, eine Leerstelle zwischen die Anführungszeichen zu tippen (" ").

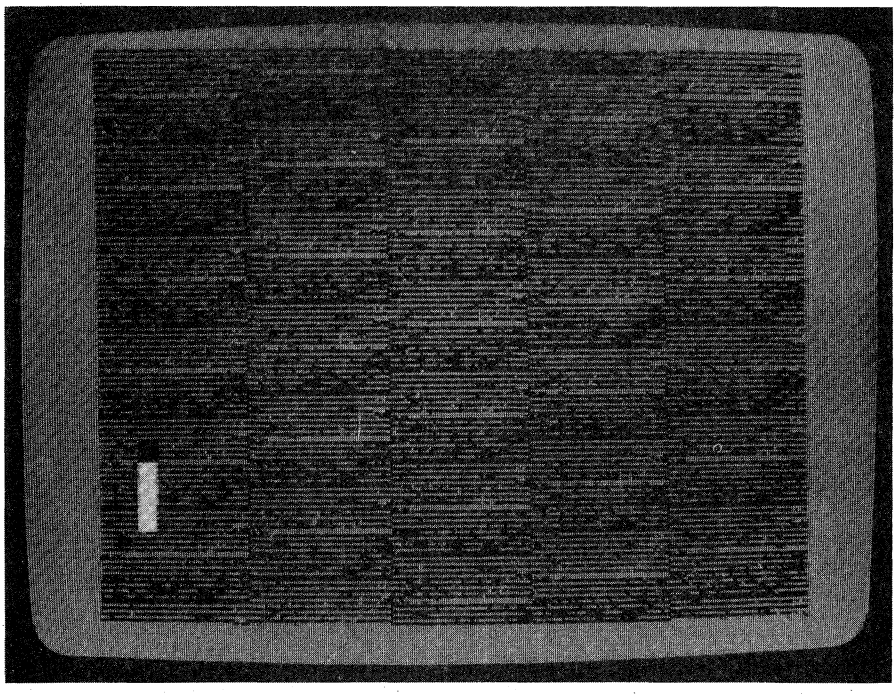
```
1100 GOSUB 20          : REM GRAPHICS
1110 C = 14: GOSUB 40: REM COLORS
1200 REM:::::::::LIGHTHOUSE
1210 POKE 18090,0: POKE 18130,17
1220 POKE 18170,17: POKE 18210,17
5000 GET A$
5010 IF A$ = " " THEN 6000
5020 GOTO 5000
6000 GOSUB 30
6010 END
```

Überprüfen Sie das neue Programm, bevor Sie es starten, auf die Richtigkeit aller Zahlen (besonders die Zeilen 1210 und 1220).

Sie sehen sofort den hochauflösenden Bildschirm. Außerdem wird der Bildschirm mit der blauen und schwarzen Vorder-/Hintergrundfarbe koloriert (selbst wenn dies schon vorher geschehen ist). Der Leuchtturm erscheint zum Schluß in Form eines senkrecht stehenden Rechtecks mit schwarzer Spitze und weißem Sockel.

Wenn Sie G, T oder C betätigen, sehen Sie, daß dies am Bildschirm nichts ändert. Betätigen Sie die LEERTASTE. Sie erhalten wieder den Textbildschirm und die Programmausführung ist abgebrochen.

Sollten Sie Probleme mit diesem Programm haben, überprüfen Sie nochmals die Zeilen 1100 bis 6010. Wenn Sie den hochauflösenden Bildschirm nicht sehen, prüfen Sie die Zeile 1100. Ändert der hochauflösende Bildschirm die Farben nicht in Blau und Schwarz, sehen Sie sich Zeile 1110 nochmals genau an. Wenn der Leuchtturm nicht erscheint, überprüfen Sie die Zeilen 1210 und 1220. Erhalten Sie den Textbildschirm nicht zurück, betätigen Sie gleichzeitig RUN/STOP und RESTORE. Überprüfen Sie die Zeilen 5010 bis 6010, besonders Zeile 5010.

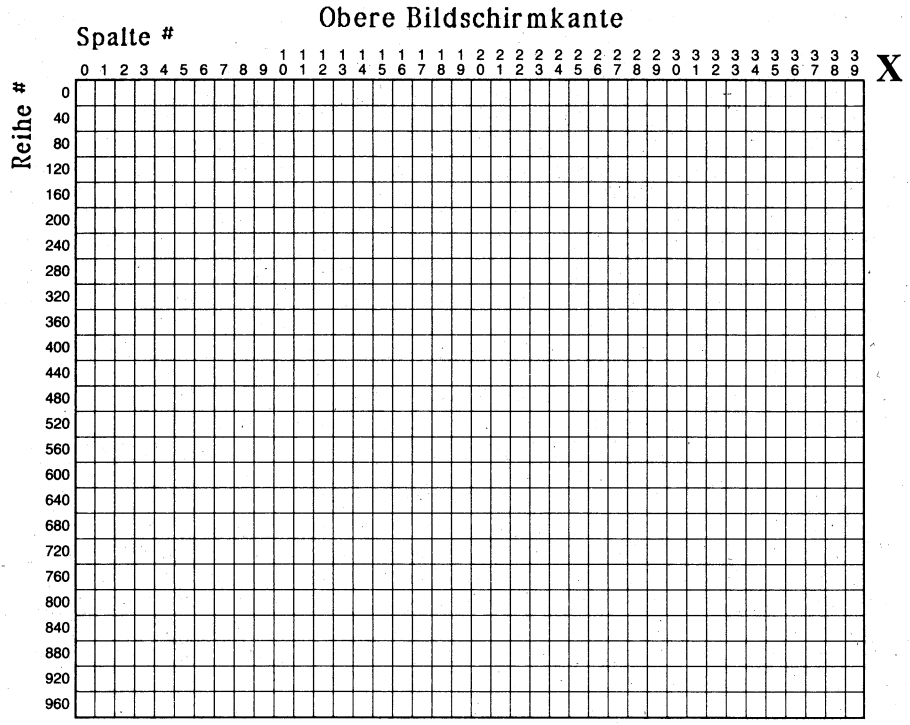


Farbenspeicher

Die Zeile 1100 wird als erste in der Hauptroutine vom Computer gelesen. Sie enthält einen GOSUB Befehl, der den Computer sofort zur Unterroutine in Zeile 20 schickt. Diese Routine schaltet auf hochauflösende Grafik um. Der Computer wartet nicht mehr darauf, daß Sie G betätigen, sondern geht direkt, wenn das Programm läuft, zur Grafik-Unterroutine über, die sofort den hochauflösenden Bildschirm wiedergibt.

Die Zeile 1110 bestimmt 14 für C. GOSUB 40 befiehlt dem Computer, die Farben-Unterroutine auszuführen, um die Vorder-/Hintergrundfarben auf dem Bildschirm zu ändern. Da 14 für Blau und Schwarz steht, erhält der Bildschirm diese Farben.

Die Zeilen 1210 und 1220 zeichnen den Leuchtturm. Wir benutzen zur Verdeutlichung das unten abgebildete Diagramm. Am einfachsten legen Sie sich Millimeterpapier griffbereit.



Zur Wiedergabe von Farbgrafik ist Ihr Bildschirm in 1000 Blöcke unterteilt, wie Sie in dem Diagramm sehen. Jeder Block enthält 64 Pixels, die Sie auf dem Bildschirm sehen. Die Blöcke sind von 0 bis 999 numeriert, von der oberen linken Ecke nach rechts fortlaufend. Jede Reihe besteht aus 40 Blöcken, jede Spalte aus 25.

Mit bestimmten Zahlen können Sie die Vorder- und Hintergrundfarbe jedes Blocks ändern. Obwohl jeder Block nur zwei Farben enthalten kann, müssen nicht alle Blöcke in denselben zwei Farben angezeigt werden, wie ja beim Einzeichnen des Leuchtturms deutlich geworden ist. Das Programm befiehlt dem Computer, die Vorder- und Hintergrundfarbe in Schwarz umzuändern (so daß der gesamte Block aus schwarzen Pixels besteht). Drei weitere Blöcke werden benötigt, um den Sockel des Leuchtturms zu bilden. Das Programm befiehlt dem Computer, jeden der Blöcke mit weißer Vorder- und

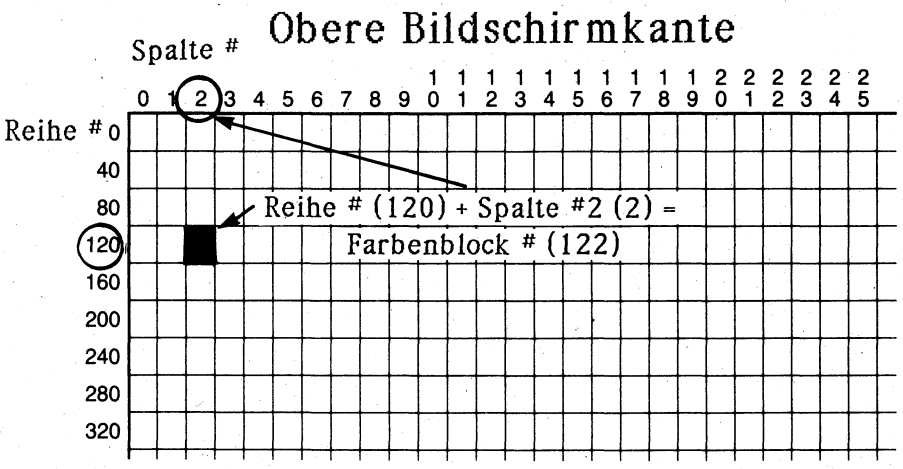
Hintergrundfarbe wiederzugeben (auch in diesem Fall hat der Block eine einheitliche Farbe).

Wie kann der Computer wissen, welche Blöcke welche Farben erhalten sollen? Sehen Sie sich die Zeilen 1210 und 1220 an. Die POKE Befehle plazieren Farbcodes (0 und 17) in die Speicherzellen 18090, 18130, 18170 und 18210. Diese Speicherstellen sollten Ihnen bekannt vorkommen. Sie gehören zur Gruppe der Speicherzellen von 17408 bis 18407, die sämtliche Farbcodes speichern. Für Ihren Leuchtturm haben Sie vier von ihnen geändert.

Mit Hilfe des oben abgebildeten Rasters und Millimeterpapier können Sie leicht Farbcodes in jeder gewünschten Speicherzelle unterbringen. Zeichnen Sie auf Ihrem Millimeterpapier ein großes Rechteck, das 40 x 25 Kästchen breit und lang ist. Markieren und Numerieren Sie die Zeichnung an Hand der obigen Abbildung.

Stellen Sie sich das Blatt als Ihren Bildschirm vor und suchen Sie den jeweiligen Block oder die Blöcke, die andere Farben erhalten sollen.

Angenommen, Sie wollen die Farben des Blockes in der dritten Spalte, vierte Reihe ändern (siehe unten). Addieren Sie die Spaltennummer (2) zur Reihennummer (120) und Sie erhalten die **Blocknummer** für diesen Block (122).



Da der erste Block von der Speicherstelle 17408 gesteuert wird, müssen Sie 17408 zu jeder Blocknummer addieren. 122 + 17408 = 17530. Um die Farben zu ändern, benutzen Sie einen POKE Befehl und einen Farbcode (z.B. POKE 17530,0, mit der dieser Block die Farbe Schwarz erhält).

Sehen Sie sich die Zeilen 1210 und 1220 an. Der Computer plaziert einen Farbcode an den Speicherstellen 18090, 18130, 18170 und 18210. Sie finden diese Stellen in Ihrer Zeichnung, indem Sie die Blocknummern ermitteln und 17408 zu jeder addieren.

In Zeile 1210 wird der Farbcode 0 für den Block 18090 eingesetzt. Auf der Farbkarte sehen Sie, daß der Farbcode 0 dem Block die Farben Schwarz auf Schwarz gibt (also einheitlich Schwarz). Die Speicherstelle in Zeile 1210, sowie die Stellen in Zeile 1220, haben alle den Farbcode 17. Die Karte zeigt, daß dieser Code weiße Pixels für den Vorder- und Hintergrund erzeugt (also einen einheitlich weißen Block).

Probieren Sie dies nun aus, indem Sie die Zeilen 1210 und 1220 ändern. Suchen Sie vier Blöcke in Ihrer Zeichnung aus. Addieren Sie zu den Nummern der Blöcke (zwischen 0 und 999) jeweils 17408. Setzen Sie diese Zahlen in die POKE Befehle der Zeilen 1210 und 1220 ein (indem Sie sie an die Stelle von 18090, 18130, 18170 und 18210 setzen). Um die Farben zu ändern, suchen Sie andere aus der Farbkarte heraus und setzen Sie die Codes hinter das Komma in den POKE Befehl. Starten Sie das Programm und sehen Sie sich das Ergebnis an.

Denken Sie daran, daß die Formel für die Änderung der Farben immer folgendermaßen lauten muß:

```
POKE 17408 + Blocknummer,Farbcode
```

Sie können entweder selbst 17408 zur Blocknummer addieren, oder den Computer für sich rechnen lassen (z.B. POKE 17408+12,0)

Wenn Sie die Änderung der Farben ausprobiert haben, geben Sie die Zeilen wieder in ihrer ursprünglichen Form ein:

```
1210 POKE 18090,0: POKE 18130,17
1220 POKE 18170,17: POKE 18210,17
```

Nun zu den letzten Zeilen (5000 bis 6010) in Ihrem neuen Programm. Die Zeile 5000 befiehlt dem Computer wieder GET A$. Wenn das Programm abläuft, zeigt der Computer nicht nur den hochauflösenden Bildschirm an, er wartet auch geduldig auf A$. Wird eine Taste betätigt, passiert zweierlei. Die Zeile 5010 befiehlt dem Computer zu Zeile 6000 zu gehen, wenn die LEERTASTE betätigt wird. Die Zeile 5020 befiehlt, zurückzugehen zu GET A$ (Zeile 5000), wenn eine andere Taste gedrückt wird. So zeigt der Computer weiter Ihr Bild an und wartet darauf, daß für A$ die LEERTASTE bestimmt wird, bevor er überhaupt die Zeile 6000 liest. Betätigen Sie schließlich die LEERTASTE, liest der Computer die Zeile 6000, geht

zur Unterroutine, die wieder den Textbildschirm anzeigt, und kehrt zur Hauptroutine zurück. Dort liest er die Zeile 6010, die das Programm beendet.

Zusammenfassung

Sichern Sie Ihr neues Programm unter "Kapitel2". (Haben Sie auch daran gedacht, die Zeilen 1210 und 1220 zu ändern? Steht in Zeile 1005 C = 14?). Sollten Sie vergessen haben, wie Sie Ihr Programm abspeichern, lesen Sie den entsprechenden Abschnitt in Kapitel 1.

In diesem Kapitel haben Sie einiges über Grafik auf dem **Commodore 64** hinzugelernt. Mit Hilfe der Grafik-Unterroutine erhalten Sie in jedem Programm den hochauflösenden Bildschirm. Mit der Farben-Unterroutine können Sie die Vorder-/Hintergrundfarben des gesamten Bildschirms verändern. Wenn Sie nur die Farben eines Bildschirmblockes ändern wollen, suchen Sie die Blocknummer (zwischen 0 und 999) in Ihrer Zeichnung, addieren 17408 zu dieser Zahl und plazieren einen Farbcode an die Speicherstelle, die mit dieser Zahl bezeichnet ist. Schließlich haben Sie noch eine Unterroutine, mit der Sie wieder den Textbildschirm erhalten, wann immer Sie dies wünschen.

Wir empfehlen Ihnen, diese notwendigen Werkzeuge immer im Gedächtnis zu behalten, wenn Sie ein Bild zeichnen, ohne sie jeweils ganz neu eingeben zu müssen. Starten Sie einfach nur die ZAP-Unterroutine mit RUN 10.

Unten sehen Sie zwei Übungsaufgaben zur Plazierung von Farben in Bildschirmblöcken. Wenn Sie wollen, lösen Sie sie. (Die Lösung finden Sie direkt unter der Aufgabe, falls Sie irgendwo steckengeblieben sind.) Sichern Sie aber zuerst, bevor Sie die Aufgaben in Angriff nehmen, das Programm aus Kapitel 2 auf Ihrer Diskette oder Ihrem Band.

Aufgabe 1

Zuerst ändern Sie die Farbe des Bildschirms in einfarbig Blau. Suchen Sie die Blöcke 2, 41, 42, 43, 82, 121 und 123 in Ihrer Zeichnung. Mit 7 POKE Befehlen ändern Sie:

1. Block 2 in einfarbig Weiß
2. die Blöcke 41, 42, 43 und 82 in einfarbig Rot
3. die Blöcke 121 und 123 in einfarbig Schwarz

Starten Sie das Programm mit RUN. Was passiert?

Lösung

Um dem Bildschirm eine einheitlich blaue Farbe zu geben, muß die Zeile 1110 " C = 102: GOSUB 40: REM COLORS " lauten. Um die Blöcke zu ändern, können Sie irgendwelche neuen Zeilennummern benutzen, aber die 7 POKE Anweisungen müssen lauten:

```
POKE 17410,17
POKE 17449,34
POKE 17450,34
POKE 17451,34
POKE 17490,34
POKE 17529,0
POKE 17531,0
```

Aufgabe 2

Zu den Zeilen, die Sie in Aufgabe #1 eingegeben haben, fügen Sie Zeilen hinzu, die den Blöcken 49, 85, 86, 87, 88, 126 und 128 die Farbe Braun geben.

Lösung

Auch hier können Sie die Zeilennummern frei wählen, aber die neuen POKE Anweisungen müssen lauten:

```
POKE 17457,153
POKE 17493,153
POKE 17494,153
POKE 17495,153
POKE 17496,153
POKE 17534,153
POKE 17536,153
```

Haben Sie die Aufgaben richtig gelöst, starten Sie das Programm mit RUN, worauf Sie die ziemlich primitive Zeichnung eines Mannes mit Hund sehen. Wenn Sie Probleme hatten, haben Sie möglicherweise die Zahl 17408 nicht zu jeder Blocknummer addiert. Diese Addition ergibt die Speicherstelle, die die Farben des Blockes steuern. Indem Sie einen Farbcode in die Speicherstelle plazieren, werden die Farben des Blockes in der gewünschten Weise geändert.

Im nächsten Kapitel lernen Sie, Punkte (Pixels) zu ermitteln und einzuzeichnen.

Kapitel 3
Ermitteln und Einzeichnen von Punkten

In diesem Kapitel lernen Sie, wie der blaue Himmel als Hintergrund und das Wasser gezeichnet werden. Den Himmel können Sie auf zwei Arten zeichnen. Einmal könnten Sie in der Unterroutine 40 C gleichsetzen mit einem Farbcode, der dem Bildschirm eine einheitlich blaue Farbe gibt. Das Problem bei dieser Methode liegt darin, daß Sie nie wissen, welche Pixels den Vordergrund und welche den Hintergrund bilden. Wenn Sie später einem Block andere Farben geben wollen, würden diese unbekannten Vorder-/Hintergrundpixels erscheinen. Deshalb gehen Sie nach einer anderen Methode vor, die ein neues Werkzeug beinhaltet. Bei dieser Methode wissen Sie immer, daß jedes Pixel in einem Block die Hintergrundfarbe des Blocks anzeigt, es sei denn Sie bestimmen etwas anderes.

Um das Wasser zu zeichnen, müssen Sie bestimmte Punkte ermitteln und dann einzeichnen. "Punkt" ist nur ein anderer Ausdruck für Pixel. Um ihn "einzuzeichnen", befehlen Sie dem Computer, ihn zur Vordergrundfarbe zu ändern - in diesem Fall zu Schwarz.

Es gibt viele Verwendungsmöglichkeiten für eingezeichnete Punkte. In dem Bild, das Sie zeichnen, wird das gesprenkelte Aussehen des Wassers erreicht, indem viele Punkte sehr dicht beieinander eingezeichnet werden. Auf dem Bildschirm nehmen Sie eher die Gesamtheit einer flächigen Form (das Wasser) als jeden einzelnen Punkt wahr. Sie können dies mit der Wahrnehmung eines Baumes und seiner Blätter vergleichen. Viele Menschen zeichnen die Blätter eines Baumes als eine einheitlich gefärbte Fläche anstelle jedes einzelnen Blattes. Oder stellen Sie sich das Bild eines Strandes vor: Sie sehen den gesamten Strand und nicht jedes einzelne Sandkorn.

Andererseits können Sie die Punkte auch weiter voneinander entfernt einzeichnen. Der Betrachter sieht dann jeden Punkt als einzelne Form. Ein Beispiel dazu wäre ein Bild mit Sternen in der Nacht, bei dem viele weiße Punkte weit voneinander entfernt auf einem dunklen Hintergrund eingezeichnet würden.

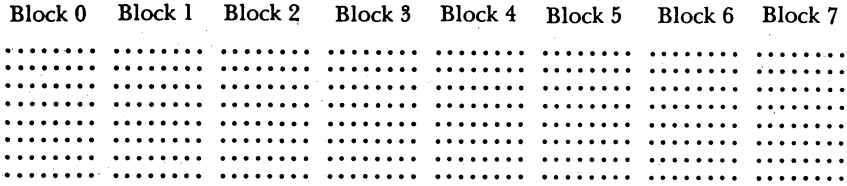
In den folgenden Kapiteln stellen wir Ihnen noch weitere Zeichen- und Farbtechniken vor, die Sie später für Ihre Bilder verwenden können. Jetzt müssen Sie erst einmal lernen, wie Sie einen der 64.000 Pixels auf dem Bildschirm markieren, damit der Computer sie einzeichnen kann.

Lokalisieren eines Punktes

Schauen Sie auf Ihren Bildschirm. Wenn Sie sehr dicht herangehen, können Sie vielleicht die einzelnen Lichtpunkte sehen, von denen insgesamt 64.000 vorhanden sind. Angenommen, jemand soll 5 dieser Punkte, alle in einer Reihe, in die Farbe Schwarz ändern. Wie würden Sie ihm sagen, welche Punkte er ändern soll (ohne sie ihm auf dem Bildschirm zu zeigen)? Sie würden die Position der Punkte beschreiben, z.B.: "Ändere die 5 Punkte in Reihe 50, Spalten 100 bis 104."

In ähnlicher Weise teilen Sie auch dem Computer mit, welche Pixels eingezeichnet werden sollen. Für eine exakte Angabe können Sie jede Spalte (insgesamt 320) und jede Reihe (insgesamt 200) auf dem Bildschirm zählen. Oder Sie schätzen - auf 2 Spalten und 2 Reihen genau. Da jede Reihe und jede Spalte sich nur um einen winzigen Punkt unterscheiden, lernen Sie schnell genau zu schätzen.

Nehmen Sie wieder ein Blatt Millimeterpapier und zeichnen Sie ein Rechteck, 40 Spalten (Kästchen) breit und 25 Reihen (Kästchen) lang. Bezeichnen Sie dieses Blatt mit "X,Y PIXEL PUNKTE". Diese Zeichnung erinnert Sie sicher an Ihr Diagramm zum Farbspeicher. Tatsächlich ähneln sich die Zeichnungen. Jede Speicherstelle in der ersten Zeichnung steuert die Farben eines Blocks aus 64 Pixels. Diese Zeichnung zeigt ebenfalls die Blöcke aus Pixels. Sehen Sie sich die Vergrößerung der ersten 8 Blöcke an, um zu sehen, wie die Pixels angeordnet sind:

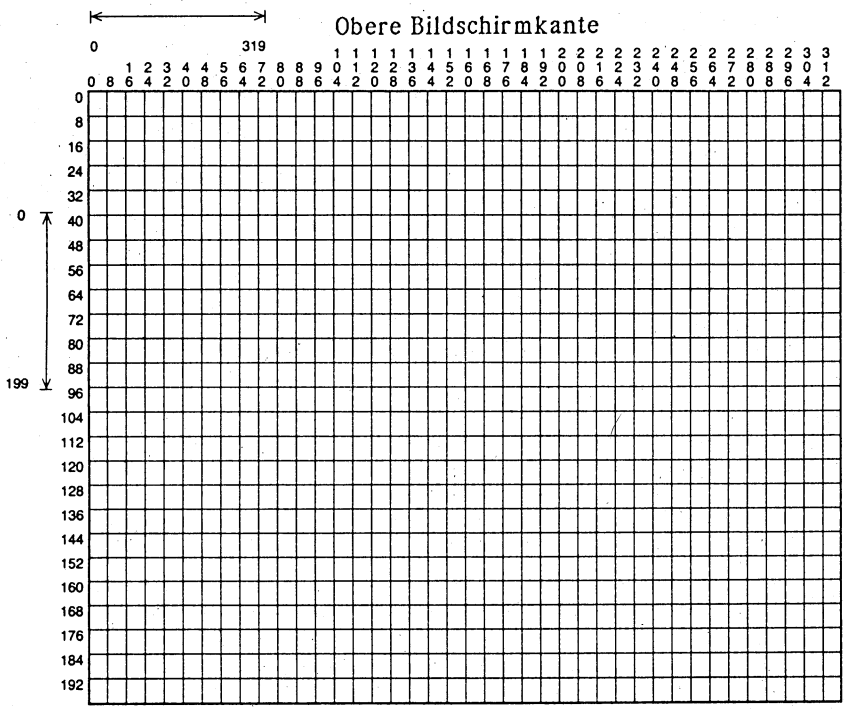


(Beachten Sie, daß auf dem Bildschirm die Blöcke **nicht** durch eine Leerstelle voneinander getrennt sind.)

Jedes Pixel auf Ihrem Bildschirm kann vom Computer lokalisiert werden, wenn Sie die Spalten- und Reihennummer angeben. Das Pixel in der oberen linken Ecke des Blocks 0 (im Zusammenhang mit Computern beginnt die Numerierung immer bei Null) wird mit "Spalte 0, Reihe 0" angegeben. Das nächste Pixel rechts befindet sich in Spalte 1, Reihe 0. Jedes Pixel in der ersten Reihe wird also mit "Reihe 0" angegeben. Das erste Pixel in der dritten Reihe befindet sich in Spalte 0, Reihe 2. Jedes Pixel in der ersten Spalte

wird also mit "Spalte 0" angegeben. Um einen Punkt einzuzeichnen, müssen Sie nur die genaue Spaltennummer (0-319) und die genaue Reihennummer (0-199) kennen, in der das Pixel sich befindet. Klingt kompliziert? Ist es auch. Es gibt eine einfachere Methode.

Anstatt 320 Spalten und 200 Reihen in Ihrer Zeichnung zu numerieren (wenn dies überhaupt möglich wäre), numerieren Sie nur das obere linke Pixel in jedem Block, wie in dem folgenden Diagramm:



X, Y Pixelpunkte

Markieren Sie in dieser Weise Ihre Zeichnung "X,Y Pixel Punkte". Nun können Sie die Stellen aller Pixels annähernd bestimmen. Angenommen, Sie zeichnen ein Bild in Ihr Diagramm und stellen fest, daß Sie ein Pixel einzeichnen müssen, das geringfügig rechts neben

Spalte 80 und etwas unter Reihe 144 liegt. Wenn Sie beispielsweise Spalte 83, Reihe 146 schätzen, liegen Sie mit einer möglichen Abweichung von etwa zwei Stellen in der Lage der Pixels richtig.

Eines müssen Sie noch wissen, wenn Sie Pixels lokalisieren. Der Computer versteht die Worte "Spalte" und "Reihe" nicht. Mit Hilfe der neuen Unterroutine versteht der Computer aber, daß X die Spalte und Y die Reihe bezeichnet. Um ein Pixel in Spalte 83, Reihe 144 einzuzeichnen, geben Sie an, daß X = 83 und Y = 144 ist. Um die schwarzen Pixels in das Wasser Ihres Bildes einzuzeichnen, geben Sie eine ganze Sammlung von X,Y Zahlen ein. Diese Sammlung von Zahlen wird als Koordinaten eines Punktes bezeichnet.

Der Hintergrund

Laden und Listen Sie das Programm aus Kapitel 2. Unten sehen Sie die neuen Zeilen, die Sie eingeben, um den Hintergrund zu zeichnen. Sie benötigen dazu nur eine Unterroutine und einen GOSUB Befehl.

```
50 REM:::::::::PAINT BKGROUND
51 FOR I = 24576 TO 32575
52 POKE I,0
53 NEXT I
54 RETURN
1120 GOSUB 50          : REM PAINT BKGROUND
```

Geben Sie jetzt diese neuen Zeilen ein. Beachten Sie, daß hinter dem Komma in Zeile 52 eine Null (0) und nicht ein O steht. In Zeile 1120 können Sie den Doppelpunkt und den REM-Befehl so weit nach rechts rücken, wie Sie wollen. Wenn Sie die Zeile in dieser Form eingeben, hebt sie sich vom Rest des Programms deutlich ab.

Wenn Sie Ihre Eingabe überprüft und Schreibfehler berichtigt haben, starten Sie das Programm. Es dauert einige Sekunden, bis auf dem Bildschirm etwas passiert. Das liegt daran, daß der Computer jedem Block die Farben Blau und Schwarz gibt, unabhängig davon, ob der Bildschirm schon Blau und Schwarz ist. Das Programm beinhaltet diesen Befehl.

Dann sehen Sie, daß jede Blockreihe die Farbe Blau erhält. Dies erfüllt einen doppelten Zweck. Erstens wird auf dem Bildschirm jedes Farbengemisch gelöscht, so daß eine einheitliche Hintergrundfarbe zurückbleibt. Zweitens wissen Sie nun ganz genau, welche Pixels Vordergrundfarbe haben (0) und welche Pixels Hintergrundfarbe haben

(64.000). Später können Sie die Farben in jedem Block ändern, ohne Überraschungen (Vordergrundpixels) zu erleben.

Betätigen Sie die LEERTASTE, um Ihr Programm wieder zu erhalten. Wenn das Programm nicht wie oben beschrieben ausgeführt wurde, haben Sie vielleicht "SYNTAX ERROR" oder eine andere Meldung auf dem Bildschirm erhalten. Fehlermeldungen geben die erste Zeilennummer an, mit der der Computer Probleme hat. (Sie können ein Programm falsch eingegeben haben und keine Fehlermeldung erhalten, verlassen Sie sich also nicht immer darauf.)

> Beachten Sie: Wenn ein Programm nicht richtig ausgeführt wird und Sie mit der Betätigung der LEERTASTE den Textbildschirm nicht zurückerhalten, geben Sie sorgfältig GOSUB 30 RETURN ein. Sie können Ihre Eingabe nicht sehen, achten Sie deshalb besonders auf die richtige Schreibweise der Bestandteile. Auf diese Weise können Sie eine Unterroutine außerhalb des Programms benutzen. Sie müßten nun eigentlich den Textbildschirm zurückerhalten und sehen möglicherweise eine Fehlermeldung. Mit der Betätigung von RUN/STOP RESTORE erhalten Sie zwar auch den Textbildschirm, gleichzeitig wird der Text auf dem Bildschirm aber gelöscht.

Aufgliederung der Hintergrund-Routine

Vergewissern Sie sich, daß Sie die Zeilen 50-54 (LIST 50-54 RETURN) und die Zeile 1120 (LIST 1120 RETURN) auf dem Bildschirm haben. Sie haben vielleicht gemerkt, daß diese Unterroutine fast identisch mit der Unterroutine in Zeile 40 ist. Sie sind tatsächlich miteinander verwandt. Die Unterroutine in Zeile 40 gibt dem Computer die Vorder-/Hintergrundfarben für die 1000 Blöcke mit Pixels auf dem Bildschirm. Die neue Unterroutine befiehlt dem Computer, das aktuelle Bildmuster zu benutzen, wenn er die Hinter-/Vordergrundfarben anzeigt. Das Bildmuster wird durch 8000 Speicherstellen bestimmt (24576 bis 32575), von denen jede 8 Pixels steuert. Die Plazierung eines Mustercodes in jeder der Speicherstellen, teilt dem Computer mit, welche Pixels Vordergrundfarbe erhalten sollen.

Jeder der 1000 Blöcke des Bildschirms besteht aus 8 Reihen mit je 8 Pixels. Für jede Reihe mit 8 Pixels ist eine Speicherstelle zuständig, die steuert, wieviele der 8 Pixels Vordergrundfarbe erhalten.

Obwohl nur eine Speicherstelle die Farben jedes Blocks steuert, steuern 8 Speicherstellen das Bildmuster jedes Blocks. Da der Bildschirm aus 1000 Blöcken besteht, steuern 8000 Speicherstellen das Bildmuster des gesamten Bildschirms. Indem Sie den Code 0 in jeden dieser Blöcke plazieren, teilen Sie dem Computer mit, daß keine (0) Pixels in jeder Reihe zu 8 die Vordergrundfarbe (Schwarz) erhalten sollen. Zurück bleibt die Hintergrundfarbe (Blau), entsprechend eingesetzt.

Diese Unterroutine ist doppelt wichtig. Erstens muß in die meisten Bilder eine Hintergrundfarbe gezeichnet werden. Dieses Werkzeug erledigt das für Sie. Zweitens löscht die Änderung aller Vordergrundpixels in Hintergrundpixels automatisch jedes vorherige Bild auf dem hochauflösenden Bildschirm. Wenn sich Vordergrundabbildungen auf dem hochauflösenden Bildschirm befinden, bleiben sie dort, bis sie durch obige Änderungen gelöscht werden oder Sie den Computer ausschalten. Wenn Sie diese Unterroutine in alle Programme einfügen, die ein Bild zeichnen, ersparen Sie sich das Aus- und Einschalten des Computers, sobald Sie ein neues Bild zeichnen wollen.

Werkzeug 50 :::::::::: Hintergrundzeichnung

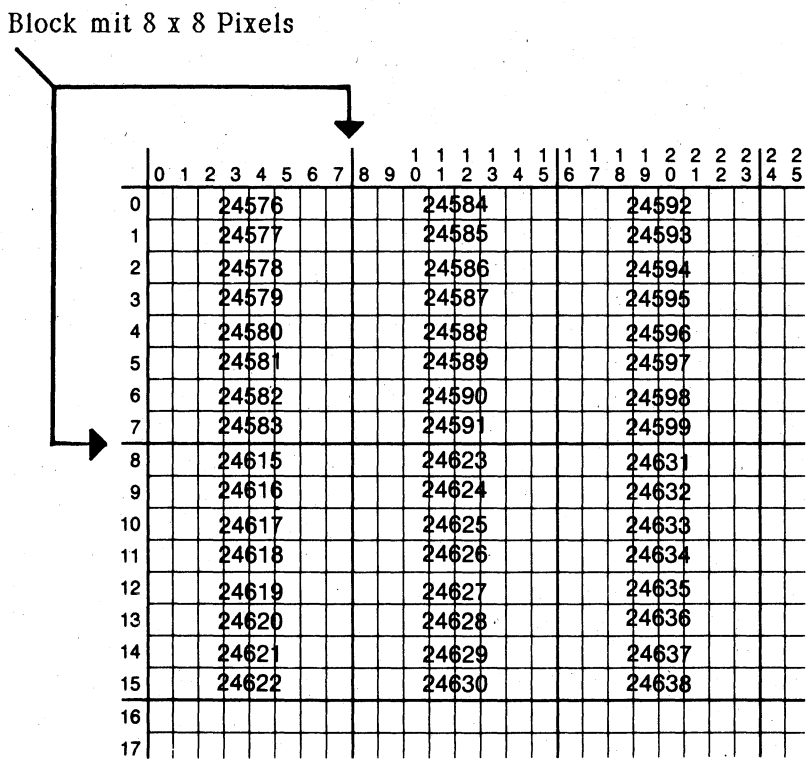
```
50 REM:::::::::PAINT BKGROUND
51 FOR I = 24576 TO 32575
52 POKE I,0
53 NEXT I
54 RETURN
```

***Zweck**: Dieses Werkzeug löscht auf dem hochauflösenden Bildschirm ein vorhandenes Bild, während es die Hintergrundfarbe zeichnet.*

***Anwendung**: Um dieses Werkzeug zu benutzen, benötigen Sie einen GOSUB Befehl in Ihrer Hauptroutine.*

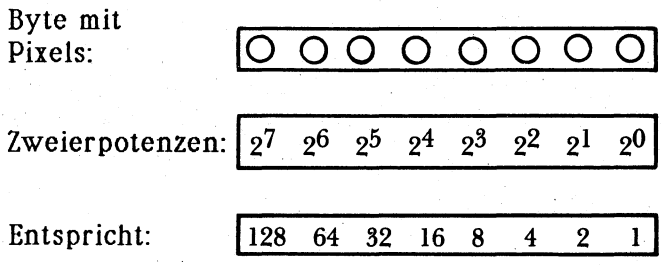
***Technische Beschreibung**: Die Speicherstellen 24576 bis 32575 bestimmen das Vorder/Hintergrund-Pixelmuster für jedes Byte Pixels auf Ihrem Bildschirm. Ein Byte ist eine Reihe von 8 Pixels in einem Block mit Pixels. Die ersten 8 Speicherstellen (24576- 24583) steuern die 8 Bytes von Pixels im oberen linken Block auf Ihrem Bildschirm. Die nächsten 8 Speicherstellen steuern die 8 Bytes von Pixels im zweiten Block der ersten Reihe. Jeder Satz von 8 Speicherstellen steuert 8 Bytes in jeder Blockreihe, den Bildschirm abwärts.*

Das unten abgebildete Diagramm zeigt einen Teil der Blöcke oben links auf Ihrem Bildschirm. In den Blöcken sind die Speicherstellen angegeben, die das Pixelmuster für jedes Byte bestimmen. Ein Byte ist eine Reihe mit 8 Pixels innerhalb eines Blocks. Die Numerierung der Speicherstellen beginnt im ersten Block oben rechts, führt acht Stellen hinunter und beginnt wieder im nächsten Block für die nächsten acht Speicherstellen.



Obere linke Ecke des Bildschirms

Um das Pixelmuster in einem Byte zu ändern, plazieren Sie einen Code an der Speicherstelle, die das Muster für das Byte bestimmt. Dieser Code gibt dem Computer genaue Anweisungen, welche Pixels Vordergrundfarbe erhalten sollen. Der Code wird durch die Addition von Zweierpotenzen bestimmt. Jedem Pixel in einem bestimmten Byte ist eine Zweierpotenz zugewiesen, beginnend mit dem Pixel ganz rechts und, wie Sie in der Abbildung unten sehen, nach links fortlaufend.



Wählen Sie die Pixels aus, die Vordergrundfarbe erhalten sollen, addieren Sie die zugehörigen Zahlen und plazieren Sie die resultierende Zahl in die entsprechende Speicherstelle. Diese Unterroutine plaziert (POKE) eine 0 in jede Speicherstelle, d.h. 0 (keine) Pixels in den 8000 Bytes sollen eine Vordergrundfarbe erhalten.

Beachten Sie, daß diese Zweierpotenzen (128, 64, 32, ..., 1) nicht von Pixel zu Pixel verschieden sind. Solange Sie die richtigen Speicherstellen angeben, benötigt der Computer keine unterschiedlichen Zweierpotenzen für die Pixel jedes Bytes.

Wenn Sie beispielsweise das erste und jedes weitere Pixel im oberen linken Byte in die Vordergrundfarbe ändern wollen, würde Ihr POKE Befehl lauten: POKE 24576,85.

Das Wasser

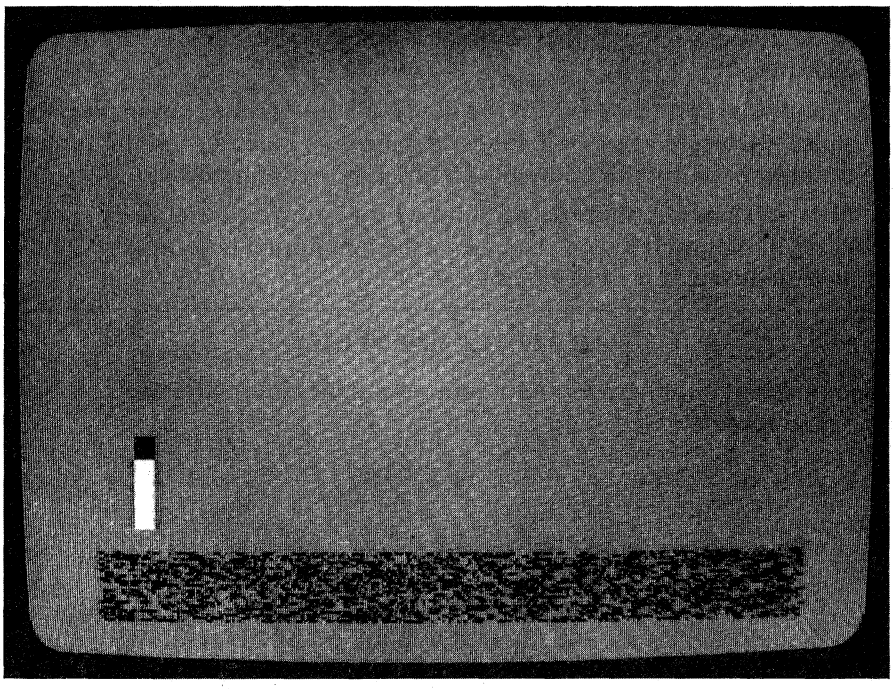
Um das Wasser zu zeichnen, müssen Sie folgende neue Zeilen für die Unterroutine eingeben:

```
60 REM:::::::::FIND A POINT
61 ROW = INT(Y/8)
62 COL = INT(X/8)
63 LINE = Y AND 7
64 BIT = 7 - (X AND 7)
65 BYTE = 24576 + ROW * 320 + COL * 8 + LINE
66 CBYTE = 17408 + ROW * 40 + COL
67 RETURN
70 REM:::::::::PLOT A POINT
71 GOSUB 60
72 POKE BYTE,PEEK(BYTE) OR 2 ^ BIT
73 POKE CBYTE,C
74 RETURN
```

Geben Sie die obigen Zeilen sorgfältig ein. Die Zeile 72 enthält das Zeichen "^". Sie finden diese Taste direkt links neben der RESTORE-Taste. Wenn Sie diese Zeilen korrekt eingegeben haben, fügen Sie Ihrem Programm die unten abgebildeten Hauptroutinezeilen hinzu.

```
1300 REM:::::::::WATER
1310 FOR Y = 176 TO 199
1320 FOR X = 0 TO 319
1330 IF RND(1) < .3 THEN GOSUB 70
1340 NEXT X: NEXT Y
```

Starten Sie das Programm, um das Endprodukt dieses Kapitels zu sehen. Der Leuchtturm wird gelöscht und der Computer zeichnet die Hintergrundfarbe (dies dauert einige Zeit). Der Leuchturm wird wieder gezeichnet, sobald der Computer zu diesen Programmzeilen gelangt. Zum Schluß sehen Sie, wie das Wasser unten auf dem



Bildschirm eingezeichnet wird. In jeder Reihe ändert der Computer 1/3 der Pixels zu Schwarz. Auf diese Weise entstehen die Schatten der blauen Wellen.

Sie müssen etwas Geduld haben, wenn der **Commodore** 64 die Punkte in Ihr Bild einzeichnet. Wenn dieser Vorgang beendet ist, betrachten Sie den Bildschirm. Langsam sieht es wie ein Bild aus!

Wenn Sie wollen, starten Sie das Programm nochmals. Danach betätigen Sie die LEERTASTE, um Ihr Programm-Listing zu erhalten.

Aufgliederung der Wasser-Routine

Listen Sie die Zeilen 1300-1340 auf dem Bildschirm und nehmen Sie sich Ihre "X,Y PIXEL PUNKTE"-Zeichnung zur Hand.

Zeile 1310 teilt dem Computer mit, daß die einzuzeichnenden Y-(Reihe) Stellen zwischen 176 und 199 liegen. Auf Ihrer Zeichnung sehen Sie, daß dies die unteren drei Blockreihen betrifft.

Zeile 1320 teilt dem Computer mit, daß die einzuzeichnenden X-(Spalte) Stellen zwischen 0 und 319 liegen. Das ist die gesamte Breite des Bildschirms. Also füllt das Wasser die unteren drei Reihen des Bildschirms in seiner ganzen Breite aus.

Wenn der Computer diesen Abschnitt des Programms liest, gelangt er zunächst zu Zeile 1310. Er setzt Y gleich mit 176. Dann gelangt er zu Zeile 1320 und setzt X gleich mit 0. So erhält der Computer die ersten X,Y-Koordinaten zum Einzeichnen (0,176).

Zeile 1330 lautet IF RND(1) < .3 THEN GOSUB 70. Wenn der Computer einen RND(1)-Befehl liest, erzeugt er intern eine Zufallszahl zwischen 0 und 1 (ausschließlich dieser beiden Zahlen). Das Resultat sind Bruchzahlen, wie .539, .686, .311, etc. Zeile 1330 teilt dem Computer mit, die Zufallszahl zu bilden und wenn (IF) sie kleiner (<) als .3 ist, zur Unteroutine in Zeile 70 zu gehen (THEN GOSUB 70), wodurch der Punkt X,Y, der in den Zeilen 1310 und 1320 bestimmt wurde, eingezeichnet wird. Durchschnittlich ist die Zufallszahl in 33% der Fälle kleiner als .3. D.h. der Computer geht in ca. 1/3 der Fälle zur Unterroutine in Zeile 70 und zeichnet den Punkt ein. Ist die Zufallszahl größer als .3, zeichnet der Computer den Punkt nicht ein. In beiden Fällen, egal ob der Computer einzeichnet oder nicht, fährt er mit der Ausführung der Zeile 1340 fort.

Der erste Befehl in Zeile 1340 lautet NEXT X. Der Computer wird zu Zeile 1320 zrückgeschickt, wo X den nächsten Wert (1) erhält. Y ist noch immer gleich 176, also lauten die Koordinaten nun 1,176. Wieder wird die Zeile 1330 gelesen und der Punkt 1,176 (in

Schwarz) eingezeichnet, wenn die erzeugte Zufallszahl kleiner als .3 ist.

Der Befehl NEXT Y wird erst dann gelesen, wenn alle Werte für X mit dem Befehl NEXT X in derselben Zeile eingesetzt wurden. Y bleibt konstant, während alle möglichen X (0-319) eingesetzt und die Punkte in 1/3 der Fälle eingezeichnet werden. Auf diese Weise werden die Punkte in Reihe 176 als erste eingezeichnet.

Wenn X zum Schluß mit 319 gleichgesetzt und der Punkt eingezeichnet wird (oder auch nicht), wird der NEXT Y Befehl gelesen. Der Computer kehrt zu Zeile 1310 zurück, wo Y mit 177 gleichgesetzt wird. Der Computer gelangt zu Zeile 1320, setzt X wieder auf 0 und beginnt mit der ersten Koordinate der nächsten Reihe (0,177). Der Wert von X wächst wieder bis 319 an, während Y konstant bei 177 bleibt.

Es handelt sich hier um einen "FOR NEXT-Befehl" innerhalb eines "FOR NEXT-BEFEHLS". Der Wert von Y bleibt solange bestehen, bis alle Werte von X bearbeitet sind. Dann erst erhält Y seinen nächsten Wert und alle Werte von X werden erneut eingesetzt. Bei jeder neuen Koordinate bestimmt Zeile 1330, ob die Unterroutine benutzt wird, um den Punkt einzuzeichnen. Dies hängt ganz von der Zufallszahl ab, die Sie als bestimmenden Faktor benutzen. Da die Zufallszahl kleiner als .3 sein muß, damit der Punkt eingezeichnet wird, werden nur 1/3 der X,Y-Koordinaten tatsächlich eingezeichnet.

Sehen wir uns nun an, was passiert, wenn die Zufallszahl kleiner als .3 ist und der Computer zur Unterroutine in Zeile 70 geschickt wird. Dazu listen Sie die Zeilen 60-74 auf dem Bildschirm.

Obwohl der Computer nun durch die Hauptroutine zu Zeile 70 geschickt wird, schickt Zeile 71 ihn eigenartigerweise sofort zu Zeile 60. Das liegt daran, daß der Computer den Punkt erst einmal auf dem Bildschirm ermitteln muß, bevor er ihn einzeichnen kann. Die Unterroutine ab Zeile 60 ist ein Werkzeug, das den Punkt auf dem Bildschirm findet. Sie fragen sich jetzt wahrscheinlich, warum die Zeile 1330 den Computer nicht direkt zu Zeile 60 schickt, wenn es die Zeile ist, zu der er zuerst gehen muß. Diese Schreibweise des Programms verhindert einfach, daß Sie irgendwann vergessen, einen Punkt zu ermitteln, bevor Sie ihn einzeichnen.

Als erstes muß der Computer die ungefähre Position des einzuzeichnenden Pixels kennen, wozu der Bildschirm in 40 x 25 Blöcke aufgeteilt ist, wie auch aus Ihrer Zeichnung ersichtlich. Anhand der Y-Koordinate findet Zeile 61 die Blockreihe, in der das Pixel sich befindet. Anhand der X-Koordinate findet Zeile 62 die Blockspalte, in der es sich befindet. So wird der Bereich, in dem das Pixel sich

befindet, erheblich eingeschränkt - von 64.000 Pixels auf nur noch 64. Die Unterroutine hat jedoch noch nicht bestimmt, welcher der 64 Pixels eingezeichnet werden soll. Zeile 64 findet die Spalte innerhalb des Blockes, in der sich das Pixel befindet. Zeile 65 sortiert die Informationen der Zeilen 61 bis 64 und liefert den genauen Block und die Reihe innerhalb des Blockes, in dem das Pixel sich befindet.

Zeile 66 erfüllt einen anderen Zweck. Sie bestimmt die Speicherstelle, die die 2 Farben des Blockes steuert, in dem der Punkt sich befindet. Sie speichert die Nummer der Speicherstelle in einem Platzhalter namens CBYTE. Sie werden gleich sehen, wie wichtig dieser Schritt ist.

Wenn dies alles erledigt ist, kann der Computer zur Unterroutine in Zeile 70 zurückkehren, um den Punkt tatsächlich einzuzeichnen. Dies geschieht in Zeile 72. Zeile 73 setzt C in CBYTE ein (POKE CBYTE,C). Wenn Sie den Wert von C ändern, bevor ein Punkt eingezeichnet wird, plaziert diese Zeile den neuen Farbcode in die entsprechende Speicherstelle, wonach sich die Hintergrundfarbe des Blockes und die Farbe des neuen eingezeichneten Punktes ändert. Denken Sie daran, daß jedes andere Pixel, das vorher in diesem Block eingezeichnet wurde, ebenfalls die neue Vordergrundfarbe erhält.

Werkzeug 60 :::::::::: Ermitteln eines Punktes

```
60 REM:::::::::FIND A POINT
61 ROW = INT(Y/8)
62 COL = INT(X/8)
63 LINE = Y AND 7
64 BIT = 7 - (X AND 7)
65 BYTE = 24576 + ROW * 320 + COL * 8 + LINE
66 CBYTE = 17408 + ROW * 40 + COL
67 RETURN
```

Zweck: *Dieses Werkzeug ermöglicht das Auffinden eines Punktes auf dem Bildschirm, um ihn dann einzuzeichnen (siehe Werkzeug 70).*

Anwendung: *Um dieses Werkzeug zu benutzen, muß die Hauptroutine die X/Y-(Spalte/Reihe-)Koordinate des zu ermittelnden Punktes angeben, z.B.:*

```
1310 Y = 180
1320 X = 10
```

Zusätzlich benötigen Sie den Befehl GOSUB 60 in Ihrem Programm. Normalerweise erscheint GOSUB 60 in der Unterroutine, die den Punkt unter vorheriger Lokalisierung einzeichnet (siehe Werkzeug 70).

Technische Beschreibung: *Wir haben bereits an anderer Stelle erklärt, wie das Programm anhand der X,Y-Koordinaten sorgfältig den Bereich eingrenzt, in dem der Punkt sich befindet. Nun können Sie schrittweise der technischen Beschreibung folgen.*

```
61 ROW  =  INT(Y/8)
```

Y kann zwischen 0 und 199 liegen. Der Computer faßt diese Reihen mit Pixels zu je acht zusammen. Das sind insgesamt 25 Blockreihen (200/8 = 25). Y/8 liefert eine Zahl zwischen 0 und 24, mit einem Rest zwischen 0 und 7. Dieser Rest ist zunächst uninteressant, da uns der BASIC-Befehl "INT" zur Verfügung steht, der den Rest entfernt und nur die ganze Zahl stehen läßt. Diese wird in der Variablen "ROW" gespeichert, bis sie später wieder benötigt wird.

```
62 COL  =  INT(X/8)
```

X kann zwischen 0 und 319 liegen. Der Computer faßt diese Spalten mit Pixels zu je acht zusammen. Das sind insgesamt 40 Blockspalten (320/8 = 40). X/8 liefert eine Zahl zwischen 0 und 39, mit einem Rest zwischen 0 und 7. Auch hier wird der Rest mit INT entfernt und die ganze Zahl in der Variablen "COL" bis auf weiteres gespeichert. Damit ist die Lokalisierung auf einen Block von 8 x 8 Pixels eingegrenzt.

```
63 LINE  =  Y AND 7
```

Obwohl Zeile 61 den Rest von Y/8 ersteinmal entfernt hat, ist er im folgenden sehr wichtig, da er besagt, in welcher Zeile das Pixel innerhalb des Blockes von 8 x 8 zu finden ist. Zeile 63 benutzt ein "Boolsche Algebra" genanntes Verfahren, um den vorher entfernten Rest zurückzuerhalten. Diese Zahl liegt zwischen 0 und 7 und ist die benötigte Blockreihe (0-7), in der das Pixel zu finden ist. Nun ist die Lokalisierung auf eine bestimmte Pixelreihe eingegrenzt.

```
64 BIT  =  7-(X AND 7)
```

Zeile 62 entfernte den Rest von X/8. Dieser Rest besagt, in welcher Reihe das Pixel innerhalb des Blockes von 8 x 8 zu finden ist. Wieder wird der Rest mit Hilfe Boolscher Algebra zurückerhalten. Die Zahl liegt zwischen 0 und 7 und ist die benötigte Spalte, in der das Pixel sich befindet. Es taucht jedoch ein kleines Problem auf, das wir vorher nicht hatten. Sie sehen die Pixels in der üblich numerierten Folge:

```
0 1 2 3 4 5 6 7
o o o o o o o o
```

der Zahlenfolge des Computers aber genau entgegengesetzt:

```
7 6 5 4 3 2 1 0
o o o o o o o o
```

Das läßt sich ebenfalls ohne große Schwierigkeiten lösen, da Zeile 64 die Spaltennummer (0-7) von 7 substrahiert, so daß alle Pixels in der Reihenfolge numeriert sind, die der Computer versteht. Das Programm beinhaltet jetzt prinzipiell alles, um den Punkt einzuzeichnen, unter der Voraussetzung allerdings, daß die Informationen aber auf vier verschiedene Variablen verteilt sind: ROW, COL, LINE und BIT.

```
65 BYTE = 24576 + ROW*320 + COL*8 + LINE
```

Diese Zeile kombiniert die Variablen ROW, COL und LINE, in denen das Pixel lokalisiert ist, und addiert die erste Speicherstelle (24576) zu dieser Summe. Das Ergebnis ist das Byte, welches das Pixelmuster an Ihrer X,Y-Koordinate bestimmt. BIT enthält noch die Pixelnummer innerhalb des Byte, die die Unterroutine in Zeile 70 benötigt, um das Pixel auf dem Bildschirm einzuzeichnen.

```
66 CBYTE = 17408 + ROW*40 + COL
```

Diese Zeile beinhaltet eine zusätzliche Information, die der Computer aus den Variablen ROW und COL erhält. Indem Sie die erste Speicherstelle für die Farben (17408) zu diesen Variablen addieren, erhält CBYTE die Farbspeicherstelle, die die Farben für das einzuzeichnende Pixel bestimmt. Vergessen Sie nicht, daß es ebenfalls die Farben der restlichen 63 Pixels des Blockes bestimmt.

Werkzeug 70 :::::::::: Einzeichnen eines Punktes

```
70 REM:::::::::PLOT A POINT
71 GOSUB 60
72 POKE BYTE,PEEK(BYTE) OR 2 ^ BIT
73 POKE CBYTE,C
74 RETURN
```

***Zweck**: Dieses Werkzeug zeichnet einen Punkt oder Punkte ein, die durch die X,Y-Koordinaten in der Hauptroutine angegeben sind. Zusätzlich plaziert dieses Werkzeug einen Code in die Speicherstelle, die die Farben der einzuzeichnenden Blöcke bestimmt, indem in die Speicherstellen die Farbencode-Variable "C" plaziert wird.*

***Anwendung**: Um dieses Werkzeug zu benutzen, muß die Hauptroutine die X/Y-(Spalte/Reihe-)Koordinate des einzuzeichnenden Punktes angeben, z.B.:*

```
1310 Y = 180
1320 X = 10
```

Außerdem muß der Punkt "gefunden werden", bevor er eingezeichnet werden kann. Dies wird mit dem GOSUB 60 Befehl vor dem Befehl GOSUB 70 erreicht oder mit GOSUB 60 innerhalb dieser Routine. Schließlich muß für "C" ein neuer Farbcode bestimmt werden, der für das Einzeichnen des Blockes benötigt wird. (Wenn Sie beispielsweise grünes Wasser vor blauem Himmel bevorzugen, hätten Sie eine Zeile 1305 mit C = 94 einfügen können.)

***Technische Beschreibung**: Zeile 71 benutzt die Unterroutine in Zeile 60, um die X,Y-Koordinaten in Ausdrücke umzuwandeln, die der Computer versteht. Bei der Rückkehr (RETURN) aus der Unterroutine 60 enthält BYTE die Bytenummer, die das Pixelmuster bestimmt und BIT die Pixelnummer, die das Pixel bestimmt. CBYTE enthält die Speicherstelle, die die Farben für den Block von 8 x 8 Pixels bestimmt, in dem sich das Pixel befindet.*

```
72 POKE BYTE, PEEK(BYTE) OR 2 ^ BIT
```

Zeile 72 zeichnet das in der Unterroutine 60 gefundene Pixel.

Sie könnten BIT in BYTE plazieren, um das Pixel einzuzeichnen. Leider bewirkt dies, daß jedes andere Pixel, das durch das Byte bestimmt wird, die Hintergrundfarbe erhält. Umgehen können Sie

dies, indem Sie zuerst nachsehen (PEEK), welche Pixels schon die Vordergrundfarbe haben - der Befehl dazu lautet PEEK(BYTE) -, um im nächsten Schritt eine sogenannte "Oderfunktion" anzuwenden, die PEEK(BYTE) mit BIT verknüpft. Sehen Sie sich zur Verdeutlichung folgendes Beispiel an:

Angenommen, die Menge der "x" in dem Pixelmuster unten sind Vordergrundpixels in einem Byte:

Pixel #: 76543210
Pixelmuster: xooooxoo

Der Computer speichert diese Information folgendermaßen:

Pixel #: 76543210
Pixelmuster: 10000100

Das "x" unten ist das Pixel, das die Vordergrundfarbe erhalten soll:

Pixel #: 76543210
Pixelmuster: oooooxooo

Der Computer speichert dies als:

Pixel #: 76543210
Pixelmuster: 00001000

Zuerst sehen Sie sich die Pixels mit der #0 an. Hat eines der beiden ein Pixelmuster 1 (d.h. Vordergrundfarbe), erhält das Pixel #0 das Pixelmuster 1. Dieser Vergleich wird bei allen Pixels ausgeführt (#0-#7), bis das endgültige Pixelmuster feststeht:

Pixel #: 76543210
Ursprüngliches Muster:
10000100
Zu änderndes Pixel:
00001000
Neues Muster: 10001100

Glücklicherweise müssen Sie diesen Vorgang nicht jedes Mal ausführen, wenn Sie einen Punkt einzeichnen wollen. Der Befehl PEEK(BYTE) OR 2 ^ BIT übernimmt das für Sie.

73 POKE CBYTE,C

Zeile 73 ändert einfach den Farbenblock mit dem eingezeichneten Punkt zu dem Farbencode in der Variablen C. (Vergessen Sie nicht, daß 63 weitere Pixels von der Änderung des Wertes von C betroffen sind.)

Zusammenfassung

In diesem Kapitel haben Sie genügend Informationen erhalten, um tatsächlich ein Bild zeichnen zu können. Sie können die Hintergrundfarben zeichnen (Speicherstellen 24576-32575) und Punkte auf dem Bildschirm ermitteln und einzeichnen. Um Zeit zu sparen, können Sie den Computer willkürlich Punkte einzeichnen lassen (soweit dies die gewünschte Wirkung erzeugt). Um einen bestimmten Punkt einzuzeichnen, geben Sie die X/Y- (Spalte/Reihe-) Koordinaten des Punktes an. Außerdem wissen Sie nun, daß das Einfügen einer neuen Programmzeile, die den Wert von C ändert, die Farben des Blockes oder der Blöcke ändert, in dem/denen eingezeichnet wird.

Drei wichtige Punkte müssen Sie sich merken:

1. X steht immer für die Spaltenkoordinate, Y für die Reihenkoordinate.
2. Sobald ein Punkt eingezeichnet ist (zur Vordergrundfarbe geändert ist), kann er die Hintergrundfarbe nur zurückerhalten, indem Sie das Programm ändern und wieder starten. Es gibt eine Unteroutine, die Punkte "löscht", die aber in diesem Buch nicht behandelt wird.
3. Sie sollten nie versuchen, einen Punkt mit negativen (-) X,Y-Koordinaten oder mit Werten größer als 319 für X und größer als 199 für Y einzuzeichnen.

Im nächsten Kapitel lernen Sie, noch mehr Zeit zu sparen. Anstatt jeden Punkt in einer Linie zu ermitteln und einzuzeichnen, werden Sie dann eine Linie durch die alleinige Angabe zweier Koordinaten ziehen können.

Unten sehen Sie zwei Aufgaben, die die neuen Informationen dieses Kapitels auswerten. Bevor Sie sich an deren Lösung versuchen, sollten Sie überprüfen, ob Sie das Programm dieses Kapitels unter dem Namen "KAPITEL 3" gesichert haben.

Aufgabe 1

Ändern Sie

1. die Zeile 1310, so daß nur die unterste Reihe der Blöcke als Wasser eingezeichnet wird
2. die Zeile 1320, so daß nur die ersten 20 Blockspalten als Wasser eingezeichnet werden
3. die Zeile 1330, so daß in 70% der Fälle, anstelle von 30%, Ihre Pixels eingezeichnet werden.

Lösung

Um nur die unterste Reihe der Blöcke einzuzeichnen, muß die Zeile 1310 lauten: FOR Y = 192 TO 199. In Ihrer Zeichnung sehen Sie, daß 192 die erste Y-Koordinate in der letzten Blockreihe ist.

Um das Wasser nur in den ersten 20 Blockspalten einzuzeichnen, muß die Zeile 1320 lauten: FOR X = 0 TO 159. In Ihrer Zeichnung sehen Sie, daß die 21. Blockspalte die Anfangskoordinate 160 für X hat. Um das Wasser nur bis zur 20. Blockspalte einzuzeichnen, wird die nächst niedrigere Koordinate 159 für X eingesetzt.

Um die Punkte in 70% der Fälle einzeichnen zu lassen, muß die Zeile 1330 lauten: IF RND(1) < .7 THEN GOSUB 70.

Aufgabe 2

Diese Aufgabe ist etwas schwieriger, aber sie zeigt bei korrekter Lösung ein hübsches Bild. Der Computer benötigt 30-45 Minuten, um das gesamte Programm zu durchlaufen, wenn Sie die Zeilen eingefügt haben. Falls Sie nicht soviel Zeit haben, nehmen Sie sich die Aufgabe später vor.

Laden Sie das Programm aus Kapitel 3 wieder in den Computerspeicher und löschen Sie die Zeilen 1210 und 1220 (Sie umgehen so das Problem, um den Leuchtturm herum einzuzeichnen). Setzen Sie C in Zeile 1110 gleich mit dem Farbencode, der für ein helles Blau als Vordergrund gegen einen schwarzen Hintergrund steht.

Schließlich geben Sie 5 neue Programmzeilen ein (1350, 1360, 1370, 1380 und 1390), die folgendes bewirken sollen:

Änderung des Farbencodes für C, dieses Mal zu weißem Vordergrund und schwarzem Hintergrund.

Bestimmung von Y und X, so daß sie am Ende die Koordinaten für alle Pixels über dem Wasser bilden.

Einzeichnen der Punkte in 1% der Fälle (.01).

Dies ist nicht ganz einfach. Wenn Sie die neuen Zeilen eingegeben haben, starten Sie das Programm. Machen Sie eine Pause, während das Programm läuft, denn das Ergebnis läßt 30-45 Minuten auf sich warten.

Lösung

Die Zeile 1110 beinhaltet C = 224. Die neuen Zeilen, die Sie hinzufügen lauten:

```
1350 C = 16
1360 FOR Y = 0 TO 175
1370 FOR X = 0 TO 319
1380 IF RND(1) < .01 THEN GOSUB 70
1390 NEXT X: NEXT Y
```

Die Zeilen 1300-1340 bleiben unverändert und das Wasser wird unten auf dem Bildschirm eingezeichnet. Da jedoch C in Zeile 1110 geändert wurde, erscheint das Wasser hauptsächlich schwarz - Schwarz ist ja nun die Hintergrundfarbe!

Zeile 1350 ändert C noch einmal, nun zu Weiß vor Schwarz. Während jeder neue Punkt eingezeichnet wird, plaziert Zeile 73 den Farbencode von C in die Speicherstelle, die die Farben des Blockes bestimmt, in dem der Punkt eingezeichnet wird.

Das Wasser beginnt in Reihe 176. Um zu verhindern, daß neue Punkte in das Wasser eingezeichnet werden, gibt Zeile 1360 175 als letzte Koordinate für Y an. Zeile 1370 gibt für X 0 bis 319 an.

Zeile 1380 zeichnet die X,Y-Koordinaten in 1% anstatt 30% der Fälle ein.

Kapitel 4
Einzeichnen von Linien und Formen

Bis jetzt haben Sie gelernt:
- die hochauflösende Grafik einzuschalten,
- den Farbcode für jeden Block auf dem Bildschirm zu ändern,
- alle Pixels auf dem Bildschirm zur Hintergrundfarbe umzukehren,
- bestimmte Punkte auf dem Bildschirm einzuzeichnen,
- zufällige Punkte auf dem Bildschirm einzuzeichnen,
- die Farben eines bereits eingezeichneten Blockes zu ändern,
- jederzeit zur Textanzeige zurüchzukehren,
- die Hauptroutine mit ZAP zu löschen, um ein neues Bild zu zeichnen.

Das ist bereits eine ganz schöne Leistung! Mit Hilfe Ihrer Werkzeuge, den Unterroutinen, haben Sie sogar schon den Teil eines Bildes erstellt: den Himmel, einen Leuchtturm und das Meer. Aber was ist mit dem Leuchtturm? Er dürfte realiter schwerlich in der Luft schweben. In diesem Kapitel ändert sich dies auch. Mit einer Unterroutine, die eine Linie einzeichnet, fügen Sie in Ihrem Bild die Umrisse einer Landzunge und Wellen ein. Mit einer Unterroutine, die Formen ausmalt, geben Sie dem Stück Land und den Wellen ihre Farben. Bevor Sie damit anfangen, geben wir Ihnen noch einige grundsätzliche Anmerkungen zum Entwurf von Vordergrundabbildungen für Ihren **Commodore 64.**

Entwerfen von Vordergrundabbildungen

Wenn Sie ein Bild auf Ihrem **Commodore 64** zeichnen wollen, sollten Sie das Bild zuerst auf Millimeterpapier skizzieren. Zeichnen Sie ein großes Rechteck, 40 Kästchen breit und 25 Kästchen lang. Skizzieren Sie Ihr Bild in diesem Rechteck. Die einzelnen Bildteile schraffieren Sie mit Buntstiften in den gewünschten Farben, die Sie jedoch nur mäßig andeuten sollten, damit die Millimetereinteilung des Papiers nicht verdeckt wird. Zum Schluß suchen Sie die Blöcke des Bildes heraus, die mehr als zwei Farben enthalten. Sie erinnern sich, daß jeder Block eine Vorder- und eine Hintergrundfarbe beinhalten kann? Wenn sich Blöcke finden, die mehr als zwei Farben haben, müssen Sie das Bild noch etwas ändern. Verlegen Sie eine Form nach

oben, unten, rechts oder links, zeichnen Sie einen kleinen Teil der Form neu oder benutzen Sie weniger Farben in dem betreffenden Teil des Bildes.

Außerdem müssen Sie beachten, daß zwei Vordergrundfarben nicht in demselben Block benutzt werden können. Um Formen im Vordergrund zu zeichnen, wird jedes Pixel in der Form eingezeichnet. Das Einzeichnen eines Pixels bedeutet eine Änderung seiner Farbe zur Vordergrundfarbe des Blockes, in dem das Pixel sich befindet. Angenommen, Sie haben zwei Formen (oder zwei Farben innerhalb einer Form), die direkt nebeneinander eingezeichnet werden müssen. Wenn diese Vordergrundfarben in einen Block fallen, gibt es keine Möglichkeit sie beide einzuzeichnen. Sie müßten den Block seperat zeichnen, als ob er eine komplette Form wäre. In der Hauptroutine würden Sie C mit den beiden Farben für diesen Block gleichsetzen (z.B. C = 94). Damit würde dann nur der Teil des Blockes, der Vordergrundfarbe beinhaltet, gezeichnet. Wenn Sie später Ihre eigenen Bilder zeichnen, werden Sie dies noch besser verstehen.

Nachdem Sie eine brauchbare Skizze fertiggestellt haben, können Sie mit den Unterroutinen die hochauflösende Grafik einschalten, die Farben des Bildschirms ändern und die Hintergrundfarbe Ihres Bildes zeichnen. Dann beginnen Sie, Punkte, Linien und Formen einzuzeichnen.

Einzeichnen von Linien

In Kapitel 3 haben Sie gelernt, daß Sie die X,Y-Koordinaten in der Hauptroutine angeben müssen, um einen einzelnen Punkt einzuzeichnen. Beispielsweise lauten die X,Y-Koordinaten des Pixels ganz rechts oben auf Ihrem Bildschirm 319,0. Wenn Punkte mit solchen X,Y-Koordinaten angegeben werden, steht der X-Wert generell vor dem Y-Wert. 0,319 wäre demnach die falsche Angabe für das Pixel oben rechts. Diese Angabe würde bedeuten, daß sich der Punkt in Spalte (X) 0, Reihe (Y) 319 befände. Wenn Sie sich Ihre Vorlage mit den X,Y PIXEL PUNKTEn ansehen, merken Sie, daß es 319 als Y-Koordinate nicht gibt. Vergegenwärtigen Sie sich immer die Formel "X vor Y", wenn wir das Einzeichnen von Linien behandeln.

Um Linien einzuzeichnen, müssen Sie nur den Anfangspunkt (X1,Y1) und den Endpunkt (X2,Y2) kennen. Angenommen, Sie wollten eine Linie von Punkt 10,20 zu Punkt 50,70 ziehen. Diese Linie wird einfach von einer entsprechenden Unterroutine und einer Hauptroutine mit den Angaben X1 = 10: Y1 = 20: X2 = 50: Y2 = 70 erstellt. Die 1 und 2 sind sehr wichtig, da sie dem Computer bestimmen, welche

X-Koordinate zu welcher Y-Koordinate gehört, um den Anfang- oder Endpunkt einer Linie zu bilden.

Wenn der Computer die Linie zeichnet, beginnt er immer am Punkt X1,Y1 und zeichnet ihn ein (ändert ihn zur Vordergrundfarbe). Dann bestimmt er sehr schnell den kürzesten, geraden Weg zum Punkt X2,Y2. In vielen Fällen ist dieser Weg nicht eine ganz gerade Linie. Sehen Sie sich die Vergrößerung eines Blockes von 8 x 8 Pixels und die eingezeichnete Linie an und Sie erkennen auf den ersten Blick warum:

```
                 oooooooo
Punkt X1, Y1 ->  --oooooo
                 oo--oooo
                 oooo--oo
                 oooooo--   <-Punkt X2, Y2
                 oooooooo
                 oooooooo     (der Buchstabe o steht für
                 oooooooo     die Hintergrundpixels,
                              und der Bindestrich - steht
                              für die eingezeichneten
                              Vordergrundpixels)
```

Da es keinen direkten Weg von Punkt X1,Y1 zu Punkt X2,Y2 gibt, muß diese Linie in einem Zick-Zack Muster eingezeichnet werden. Die Linie wird nach dem Schema "zwei zeichnen, ein Pixel nach unten, ein Pixel überschlagen; zwei zeichnen, ein Pixel nach unten, ein Pixel überschlagen; etc." eingezeichnet. Die Unterroutine berechnet das notwendige Schema und zeichnet die Linie ein.

Um mehr über das Einzeichnen von Linien zu lernen, laden Sie das Programm aus Kapitel 3. Starten Sie die ZAP-Routine (geben Sie RUN 10 ein und betätigen Sie RETURN, so daß Sie eine neue Hauptroutine eingeben können. Geben Sie zuerst die folgende Unterroutine ein:

```
80 REM:::::::::PLOT A LINE
81 DX = X2 - X1: DY = Y2 - Y1
82 L = ABS(DX): IF ABS(DY) > L THEN
   L = ABS(DY)
83 IF L > 0 THEN XI = DX/L: YI = DY/L
84 X = X1 + .5: Y = Y1 + .5
85 FOR I = 0 TO L
86 GOSUB 70: REM PLOT A POINT
```

```
87 X = X + XI: Y = Y + YI
88 NEXT I
89 RETURN
```

Sehen Sie sich die eingegebenen Zeilen noch einmal an. Vergleichen Sie die eingegebenen Zeilen mit denen, die Sie oben sehen. Wenn ein Programm nicht richtig läuft, ist es oft schwierig, herauszufinden, ob das Problem in der Hauptroutine oder einer Unterroutine liegt. Nehmen Sie sich die Zeit, jede Zeile zu überprüfen und Sie ersparen sich eine Menge Kopfschmerzen.

Für den Anfang zeichnen Sie einfach eine Linie von der oberen linken Ecke in die untere rechte Ecke des Bildschirms. Diese diagonale Linie ist leicht zu erkennen und Sie bemerken sofort, wenn sie nicht in der gewünschten Weise gezeichnet wird. Wie schon gesagt erfordert das Einzeichnen einer Linie eine X1,Y1- und eine X2,Y2-Koordinate. Nehmen Sie erneut Ihre Zeichnung X,Y PIXEL PUNKTE und versuchen Sie die notwendigen Koordinaten für diese diagonale Linie zu finden. Die richtigen Koordinaten lauten: für X1,Y1 0,0 (X1 = 0, Y1 = 0) und für X2,Y2 319,199 (X2 = 319, Y2 = 199). Geben Sie nun die folgenden Zeilen der Hauptroutine ein. Sie sehen, daß die obigen Koordinaten in dieser Routine erscheinen.

```
1100 GOSUB 20          : REM GRAPHICS
1110 C = 1: GOSUB 40: REM COLORS
1120 GOSUB 50          : REM PAINT BKGROUND
2100 REM:::::::::LINE TEST
2110 X1 = 0: Y1 = 0
2120 X2 = 319: Y2 = 199: GOSUB 80
2150 END
```

Bevor Sie das Programm starten, überprüfen Sie die Zeilen 2110 und 2120. Vergewissern Sie sich, daß X gleich 0 (X1) bzw. gleich 319 (X2) und Y gleich 0 (Y1) bzw. gleich 199 (Y2) ist. Liegt eine der X,Y-Koordinaten außerhalb beider Bereiche (0-319,0-199), könnten sich daraus Probleme ergeben. Mit "Problemen" meinen wir, daß der Computer seine Mitarbeit verweigert. Sie müßten ihn aus- und wieder einschalten, um die falsch gezeichneten Punkte aus dem Speicher zu löschen, was jedoch unweigerlich das Löschen des gesamten Programmes zur Folge hat. Wenn Sie sich etwas aus diesem Buch wirklich gut merken wollen, dann die Bereiche für die X,Y-Koordinaten.

Starten Sie das neue Programm.

Alle Hintergrundpixels werden damit zu Weiß geändert. Anschließend werden alle Vordergrundpixels zu Hintergrundpixels (also eben-

falls weiß). Zum Schluß wird die diagonale Linie von der oberen linken Ecke zur unteren rechten Ecke eingezeichnet. Sehen Sie das Zick-Zack Muster? Es gibt keinen direkten Weg von Punkt 0,0 zu Punkt 319,199. Ist es nicht trotzdem eine feine Sache, daß die Unterroutine den kürzesten, schnellsten Weg ermittelt?

Die Änderung der Farben auf dem Bildschirm in Zeile 1110 hat zur Folge, daß Sie nun eine hellblaue Rahmung um den hochauflösenden Bildbereich sehen. Sie haben es vielleicht nicht gewußt, aber diese Umrandung war immer auf Ihrem Bildschirm. Sie umgibt den hochauflösenden Bildbereich, gehört aber nicht dazu. Es gibt wirklich keine Möglichkeit darauf zu zeichnen oder sie loszuwerden. Sie haben die Umrandung bisher deshalb nicht gesehen, weil der Hintergrund auf Ihrem Bildschirm auch immer hellblau war. So gingen Bildbereich und Umrandung ineinander über und bildeten eine einheitlich gefärbte Fläche. Sie können die Farbe der Umrandung ändern, indem Sie einen Farbcode in die Speicherstelle 53280 plazieren. Beachten Sie jedoch, daß nur die Hintergrundfarbe des Codes als Umrandung benutzt wird, während die Vordergrundfarbe ignoriert wird.

Schauen Sie zu, während der Computer die diagonale Linie von oben links nach unten rechts einzeichnet. Wenn die Linie irgendwoanders oder überhaupt nicht eingezeichnet wird (warten Sie ruhig einige Minuten ab, um ganz sicher zu gehen), betätigen Sie gleichzeitig RUN/STOP und RESTORE. Überprüfen Sie in der Zeile 1110, ob C gleich 1 ist. Es könnte sein, daß der Vorder- und der Hintergrund dieselbe Farbe haben. Das würde bedeuten, daß der Computer zwar die Linie einzeichnet, Sie sie aber nicht auf dem Bildschirm sehen können. Wenn Sie für C beispielsweise den Code 17 eingesetzt hätten, würde es ausgesehen haben, als ob die Linie nicht eingezeichnet worden wäre, da 17 der Code für weißen Vordergrund vor weißem Hintergrund ist. Die Linie hätte sich vom Hintergrund nicht abgehoben, sondern wäre mit ihm verschmolzen.

Vergewissern Sie sich, daß Ihr Programm richtig läuft, bevor Sie fortfahren. Wenn die Zeilen der Hauptroutine korrekt eingegeben sind, die Linie aber nicht eingezeichnet wird, überprüfen Sie die Unterroutine ab Zeile 80. Diese Unterroutine muß exakt eingegeben sein, bevor Sie zur nächsten Übung weitergehen können.

Wenn alles überprüft ist und die diagonale Linie eingezeichnet ist, betätigen Sie RUN/STOP und RESTORE. Sehen Sie sich die Hauptroutine an. Zeile 1100 schaltet die hochauflösende Grafik ein, so daß Sie beobachten können, wie die Linie eingezeichnet wird. Zeile 1110 zeichnet die Vorder-/Hintergrundfarben auf den Bildschirm (Schwarz vor Weiß). Zeile 1120 ändert alle Pixels zur Hintergrundfarbe Weiß. (Diese Zeilen kennen Sie schon.)

Zeile 2110 gibt die Anfangskoordinaten (0,0) für die Linie an. Zeile 2120 gibt die Endkoordinaten (319,199) für die Linie an und schickt den Computer zu der Unterroutine, die die Linie einzeichnet. Sobald Sie dem Computer die X1,Y1- und X2,Y2-Koordinaten der einzuzeichnenden Linie gegeben haben, müssen Sie ihn nur noch zu der entsprechenden Unterroutine schicken. Zeile 2150 "beendet" das Programm. Diese Zeile ist nur dann notwendig, wenn der Computer die Ausführung des Programms bei einer bestimmtem Zeilennummer abbrechen soll. Wenn hinter dem Befehl END keine weiteren Zeilennummern folgen, ist dieser Befehl nicht notwendig. Obgleich sich im Moment keine weiteren Zeilen hinter dem Befehl END befinden, werden Sie später noch welche hinzufügen. Dann wird auch der Zweck des Befehls END erklärt.

Als nächste Übung wollen wir Ihnen vermitteln, an welcher Stelle zwei Linien einzuzeichnen sind. Sie überlegen sich, welche Zeilen in der Hauptroutine nötig sind, um sie einzuzeichnen. Fertig? Ändern Sie die Zeilen 2110 und 2120, um eine Linie quer in der ersten Reihe des Bildschirms zu ziehen. Darüberhinaus fügen Sie die Zeilen 2130 und 2140 für eine vertikale Linie in der äußersten rechten Spalte ein. Vergewissern Sie sich, daß Sie diese Linien nicht außerhalb des Bereiches der möglichen X- (0-319) und Y- (0-199) Koordinaten einzeichnen.

Wenn Sie glauben, daß alles richtig ist, starten Sie das Programm wieder. Die beiden Linien müßten in Schwarz oben/horizontal und rechts/vertikal eingezeichnet werden. Benutzen Sie RUN/STOP und RESTORE, um die Ausführung des Programms zu unterbrechen, falls Probleme auftauchen. Wenn der Computer überhaupt nicht mehr arbeitet und RUN/STOP RESTORE hilft Ihnen nicht weiter, haben Sie vielleicht außerhalb des möglichen X,Y-Bereiches eingezeichnet. Dann können Sie den Computer nur noch aus- und wieder einschalten, das Programm aus Kapitel 3 laden, die Unterroutine, die die Linie einzeichnet, und die Zeilen der Hauptroutine neu eingeben.

Wenn die Linien richtig eingezeichnet wurden, lauten Ihre neuen Programmzeilen:

```
2110 X1 = 0: Y1 = 0
2120 X2 = 319: Y2 = 0: GOSUB 80
2130 X1 = 319: Y1 = 0
2140 X2 = 319: Y2 = 199: GOSUB 80
```

Nach jeder neuen Bestimmung der X1,Y1- und X2,Y2-Koordinaten muß ein GOSUB 80 eingefügt werden, damit die Linie eingezeichnet

wird. Da Sie zwei Linien gezeichnet haben, haben Sie folglich zweimal den Befehl GOSUB 80 anwenden müssen.

Die Zeile 2110 gibt X1 = 0 und Y1 = 0 an. Dieser Punkt 0,0 ist das oberste linke Pixel auf dem Bildschirm. Die Zeile 2120 gibt X2 = 319 und Y2 = 0 an. Dieser Punkt ist das obere rechte Pixel auf dem Bildschirm. GOSUB 80 bewirkt, daß zwischen diesen beiden Punkten eine Linie eingezeichnet wird. Die Zeilen 2130 und 2140 zeichnen eine Linie zwischen der oberen und der unteren rechten Ecke des Bildschirms ein. So einfach ist das Einzeichnen von Linien!

Beachten Sie, daß in den Zeilen 2120 und 2130 die X,Y-Koordinaten dieselben sind. Der einzige Unterschied besteht darin, daß in Zeile 2120 X2, Y2 und in Zeile 2130 X1, Y1 steht. In der nächsten Übung lernen Sie, dieselben Koordinaten mehrmals, ohne Neueingabe zu benutzen.

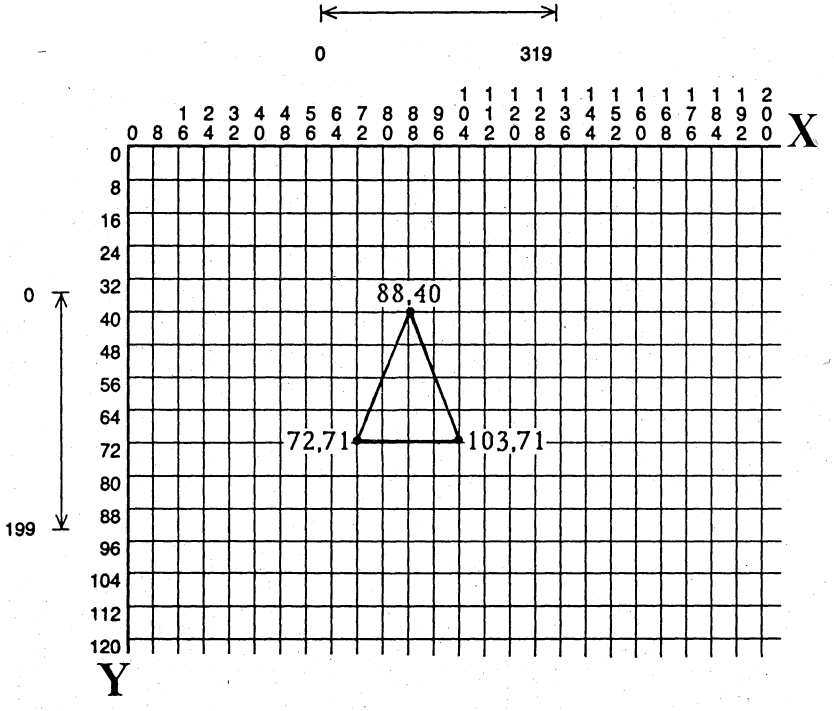
Nehmen Sie sich nun die Zeit, eigene Linien zu zeichnen. Skizzieren Sie sie zuerst in Ihrer Zeichnung X,Y PIXEL PUNKTE. An einem Ende der Linie (es ist egal, an welchem) notieren Sie die X,Y-Koordinaten des Anfangspunktes (X1 = ?, Y1 = ?), am anderen Ende die Koordinaten des Endpunktes (X2 = ?, Y2 = ?). Geben Sie diese neuen Koordinaten in die LINE TEST Routine ein (denken Sie an den möglichen Bereich für X und Y). Nachdem Sie das Programm neu gestartet haben, vergleichen Sie die neuen Zeilen auf dem Bildschirm mit denjenigen Ihrer Skizze.

Einzeichnen von Formen

Wenn mehrere aufeinandertreffende Linien gezeichnet werden, erhalten Sie die Umrisse einer Form. Beispielsweise könnten drei zu einem Dreieck verbundene Linien die Umrisse eines Hausdaches bilden. Wenn die Umrisse eingezeichnet sind, könnte der gesamte Bereich innerhalb der Umrisse schwarz oder weiß ausgemalt werden, um das Dach zu vollenden. Die Umrisse einer Form sind nicht unbedingt die eines gesamten Bildteiles oder des ganzen Bildes. Stattdessen säumen sie jeden farbigen Bereich, der gezeichnet werden soll. Wichtig dabei ist, daß die Umrisse einer Form zuerst gezeichnet werden müssen, bevor die so abgegrenzte Fläche ausgemalt wird. Wenn Sie mit der Unterroutine PLOT A LINE zurechtgekommen sind, können Sie auch mit Leichtigkeit Umrisse zeichnen.

Nehmen wir ein Dreieck. Ein Dreieck besteht aus drei aufeinandertreffenden Linien, die einen Umriß bilden. Um die Linien einzuzeichnen, benötigt der Computer zwei Koordinaten für jede von

ihnen: die Anfangs- und Endkoordinate. Zeichnen wir die Umrisse des Dreiecks ein, das Sie unten skizziert sehen.



Eine Möglichkeit, die Linien einzuzeichnen, wäre:
(Geben Sie diese Programmzeilen nicht ein)

```
3100 REM::::::::::SHAPE TEST
3110 X1 = 72: Y1 = 71
3120 X2 = 88: Y2 =40: GOSUB 80
3130 X1 = 88: Y1 = 40
3140 X2 = 103: Y2 = 71: GOSUB 80
3150 X1 = 103: Y1 = 71
3160 X2 = 72: Y2 = 71: GOSUB 80
```

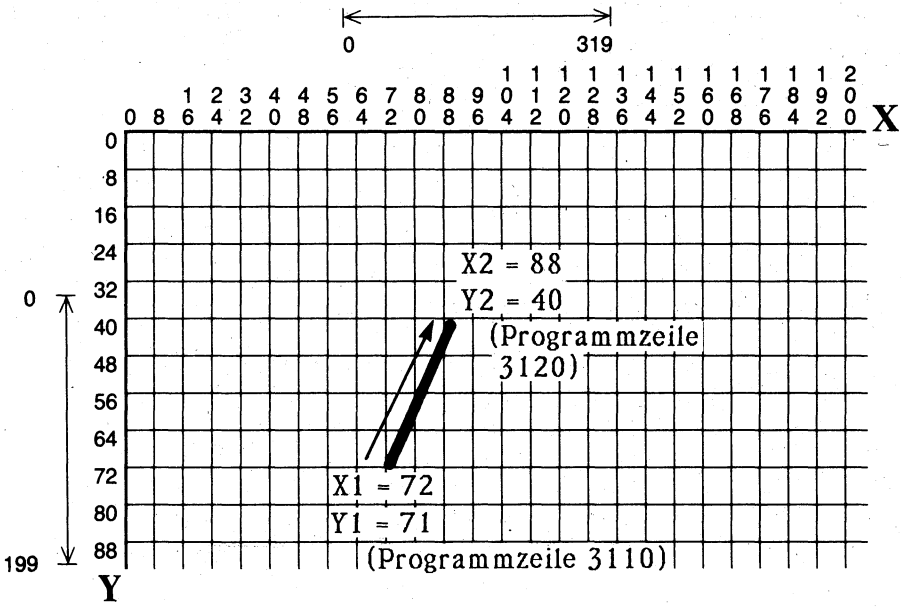
Dieses Programm würde zwar seinen Zweck erfüllen, aber Sie sehen, wie oft Sie eine schon eingegebene Koordinate wiederholen müssen, falls Sie die nächste Linie einzeichnen möchten. Mit den folgenden Programmzeilen erreichen Sie dasselbe mit weitaus weniger Arbeitsaufwand:

```
3100 REM:::::::::SHAPE TEST
3110 X1 = 72: Y1 = 71
3120 X2 = 88: Y2 =40: GOSUB 80
3130 X1 = 103: Y1 = 71: GOSUB 80
3140 X2 = 72: Y2 = 71: GOSUB 80
```

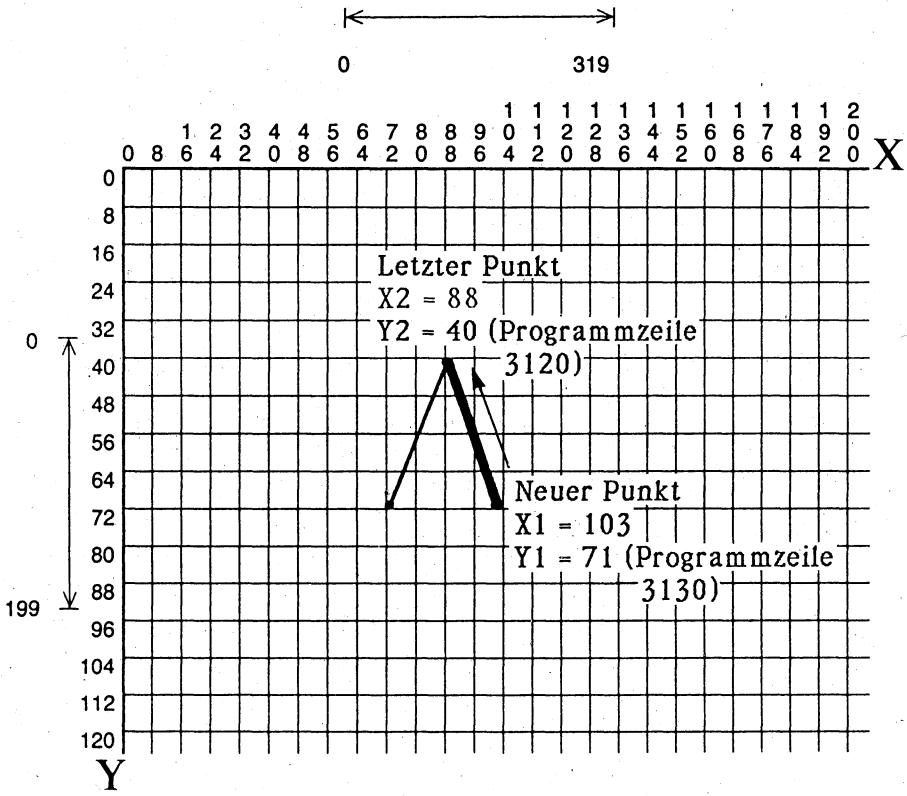
Die zweite Methode benötigt nur 5 Programmzeilen anstelle von 8; aber was bewirken diese Programmzeilen?

Wenn der Computer zur Unteroutine PLOT A LINE geschickt wird, zeichnet er immer eine Linie zwischen der zuletzt angegebenen X1,Y1-Koordinate und der zuletzt angegebenen X2,Y2-Koordinate. Es spielt keine Rolle, ob eine dieser Koordinaten schon einmal benutzt worden ist.

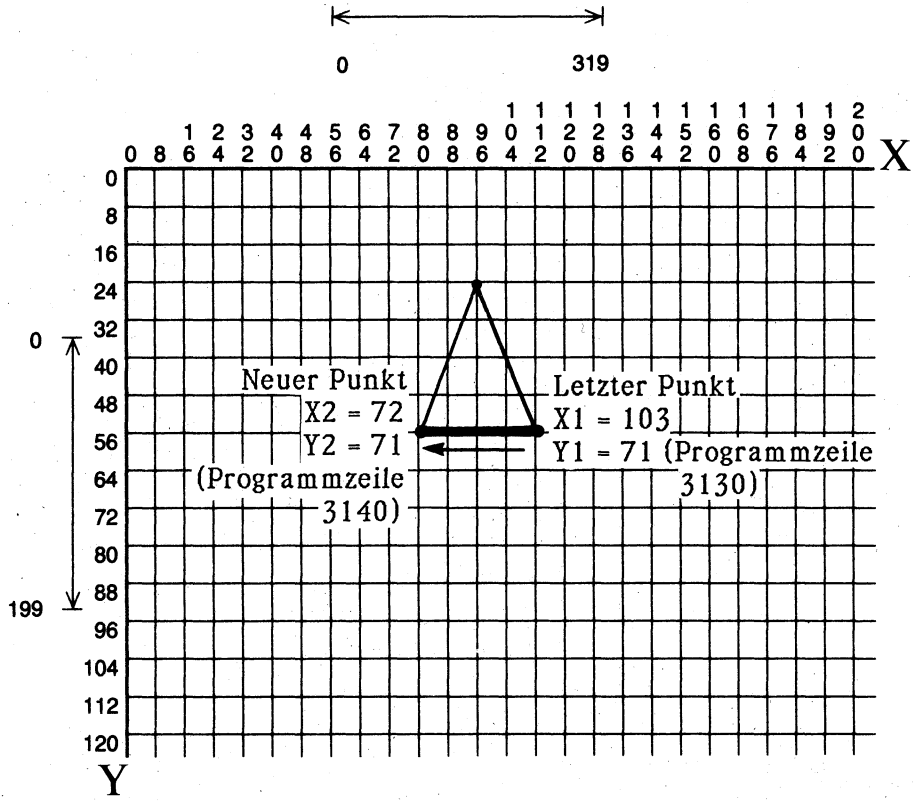
Sehen Sie sich die oben angeführten Zeilen an. Die Zeilen 3110 und 3120 zeichnen die erste Linie ein:



Wenn der Computer Zeile 3130 erreicht, sieht er, daß sich nur die X1,Y1-Koordinaten geändert haben. Er geht zur Unterroutine und zeichnet von der neuen X1,Y1-Koordinate (103,71) zur alten X2,Y2-Koordinate (88,40):



Zeile 3140 ändert X2 und Y2. Anhand der letzten X1,Y1-Koordinate des Programms (103,71) und der neuen X2,Y2-Koordinate (72,71) wird die letzte Linie eingezeichnet:

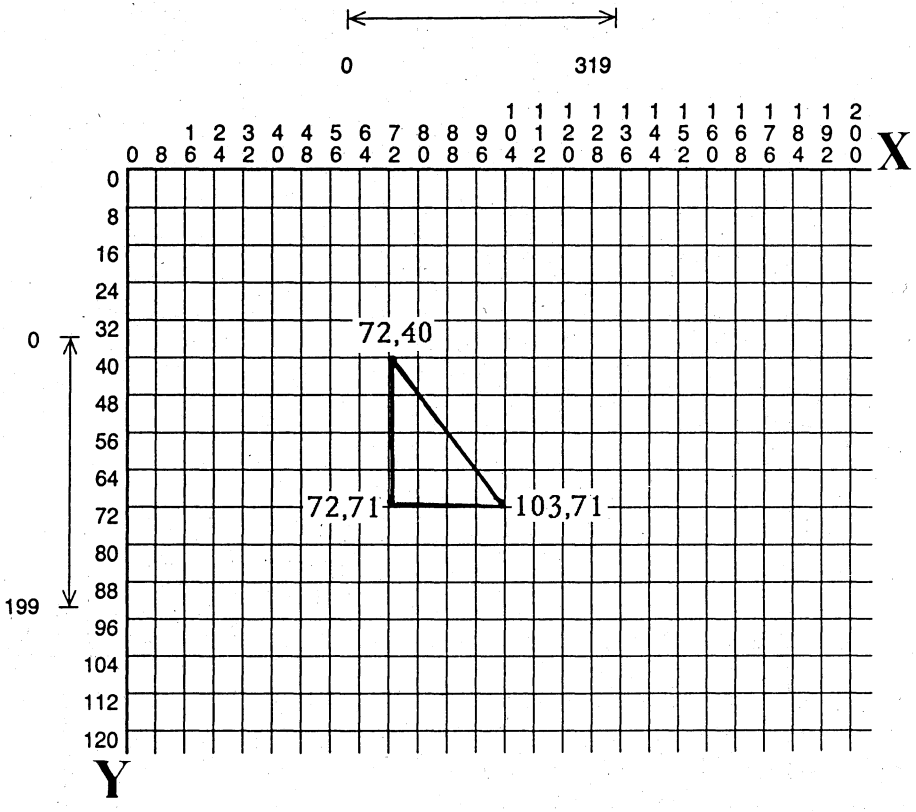


Solange der Computer ein X1,Y1- und ein X2,Y2-Paar hat, zeichnet er eine Linie ein. Er verfährt immer vom zuletzt angegebenen Punkt X1,Y1 zum zuletzt angegebenen Punkt X2,Y2 (vorausgesetzt, diesen Angaben folgt jeweils der Befehl GOSUB 80).

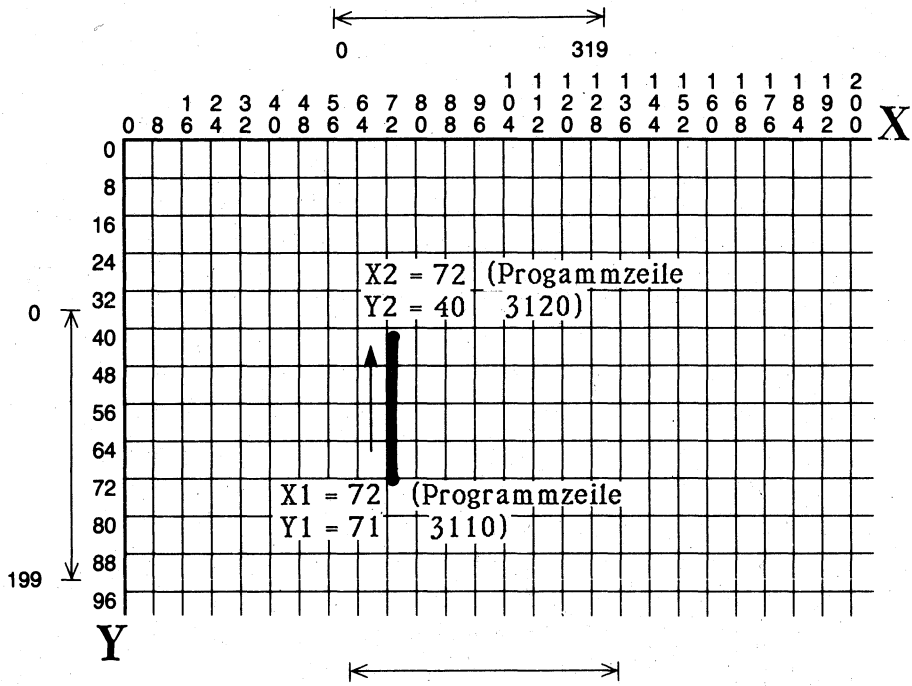
Geben Sie die oben angeführten Zeilen 3100 bis 3140 ein. Fügen Sie zusätzlich die Zeile 1130 GOTO 3100 ein (so umgeht der Computer bei der Programmausführung die Zeilen LINE TEST). Starten Sie das Programm. Beachten Sie, daß das Dreieck nicht in einem Durchgang gezeichnet wird, da jede Linie von Punkt X1,Y1 zu Punkt X2,Y2 gezogen wird. Sehen Sie sich die Skizzen dieses Dreiecks noch einmal auf die Positionen der Punkte hin an.

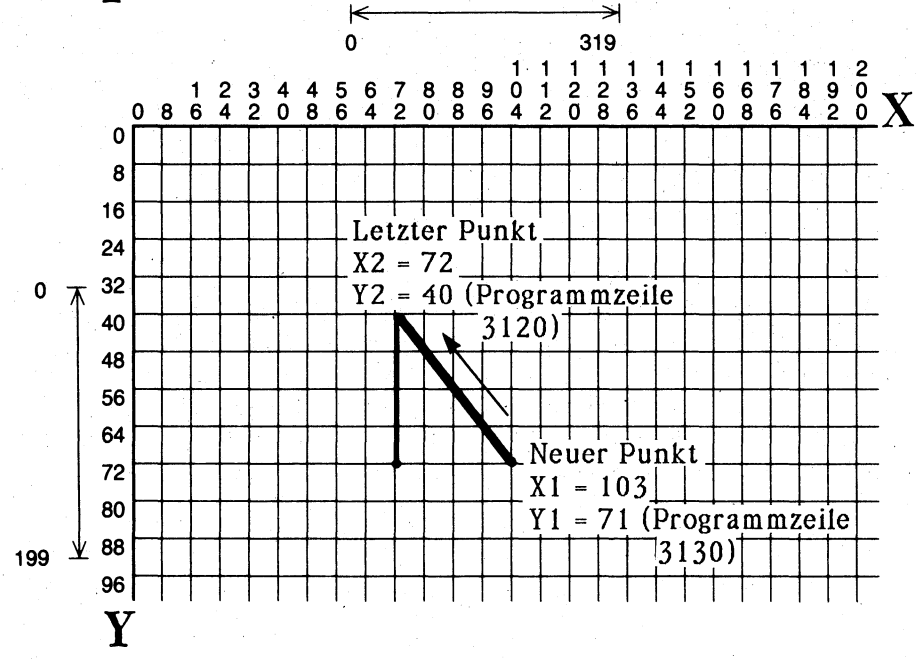
Wenn das Dreieck fertig ist, kehren Sie zum Programm-Listing zurück. Ändern Sie die Hauptroutine, um das unten abgebildete rechtwinklige Dreieck einzuzeichnen.

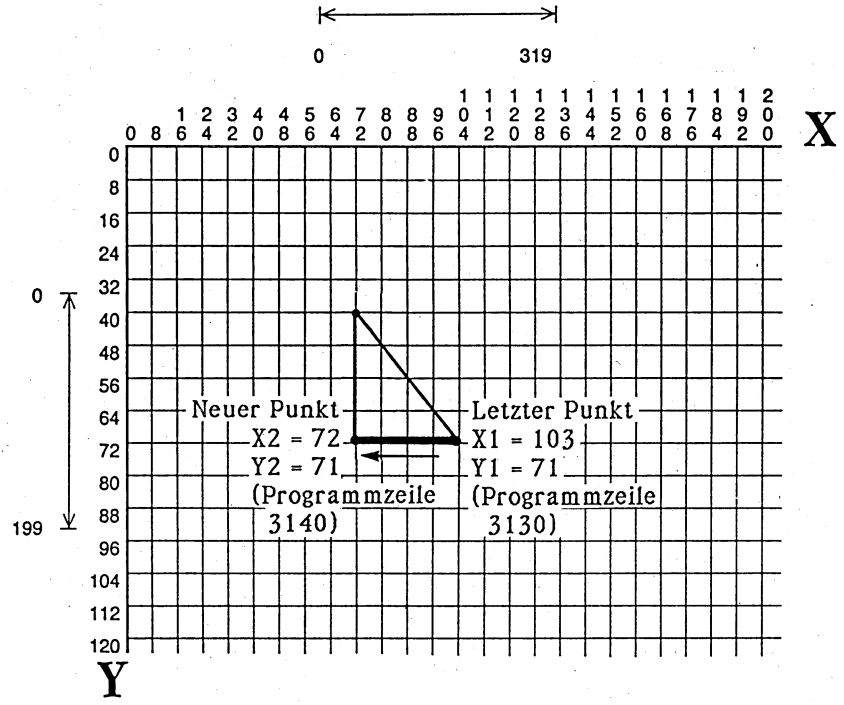
Die Lösung finden Sie im nächsten Abschnitt, aber versuchen Sie es zunächst ohne Hilfestellung. Dann starten Sie das Programm, um zu sehen, ob es funktioniert.



Um das rechtwinklige Dreieck zu zeichnen, muß nur die Zeile 3120 geändert werden. Sie muß nun lauten: 3120 X2 = 72: Y2 = 40. Diese neue Zeile verändert den Endpunkt der zuerst eingezeichneten Linie und den Anfangspunkt der als zweites eingezeichneten Linie. Sehen Sie sich zum besseren Verständnis der Programmausführung die drei nächsten Abbildungen an.


0
319
0 8 16 24 32 40 48 56 64 72 80 88 96 104 112 120 128 136 144 152 160 168 176 184 192 200
X
0
8
16
24
32
40
48
56
64
72
80
88
96
0
199
Y
X2 = 72 (Progammzeile
Y2 = 40 3120)
X1 = 72 (Programmzeile
Y1 = 71 3110)


0
319
0 8 16 24 32 40 48 56 64 72 80 88 96 104 112 120 128 136 144 152 160 168 176 184 192 200
X
0
8
16
24
32
40
48
56
64
72
80
88
96
0
199
Y
Letzter Punkt
X2 = 72
Y2 = 40 (Programmzeile
3120)
Neuer Punkt
X1 = 103
Y1 = 71 (Programmzeile
3130)



Belassen Sie diese neuen Programmzeilen in Ihrer Hauptroutine. Der nächste Abschnitt behandelt das Ausmalen der Formen, wobei das Dreieck die erste Form dazu sein wird.

Um eine größere Sicherheit im Zeichnen irgendwelcher Formen zu erlangen, üben Sie, selbst zu zeichnen und ändern Sie die Zeile 1130 zu GOTO 4100. Beginnend bei dieser Programmzeile geben Sie neue Zeilen für Ihre Formen ein. Wenn eine Linie in einer Form nicht korrekt eingezeichnet wird, gehen Sie folgendermaßen vor:

1. Listen Sie das Programm auf dem Bildschirm;
2. Beginnen Sie bei dem GOSUB 80 Befehl, der das Einzeichnen der falschen Linie bewirkt hat;
3. Verfolgen Sie das Programm zurück und notieren Sie die letzten X1-,Y1-,X2- und Y2- Koordinaten vor dem Befehl GOSUB 80;
4. Vergleichen Sie diese Koordinaten mit denen der entsprechenden Linie in Ihrer Skizze.

Um stattdessen das Einzeichnen einer einzelnen Linie zu üben, löschen Sie die Zeile 1130 völlig. Der Computer geht dann direkt beim Start des Programms zur Routine LINE TEST. Der Befehl END in

Zeile 2150 beendet das Programm, bevor die Routine SHAPE TEST beginnt. Mit der Verwendung der Befehle END und GOTO verfügen Sie über eine einfache Methode, bestimmte Teile Ihres Programms zu testen, ohne daß das Programm immer wieder ganz ausgeführt wird.

Ausmalen von Formen

Das Ausmalen von Formen ist eines der lohnenderen Bestandteile der Computergrafik. Ihre Bilder werden realistischer und für den Betrachter interessanter, wenn sie mit Farben ausgemalt werden. Geben Sie die folgende Unterroutine ein:

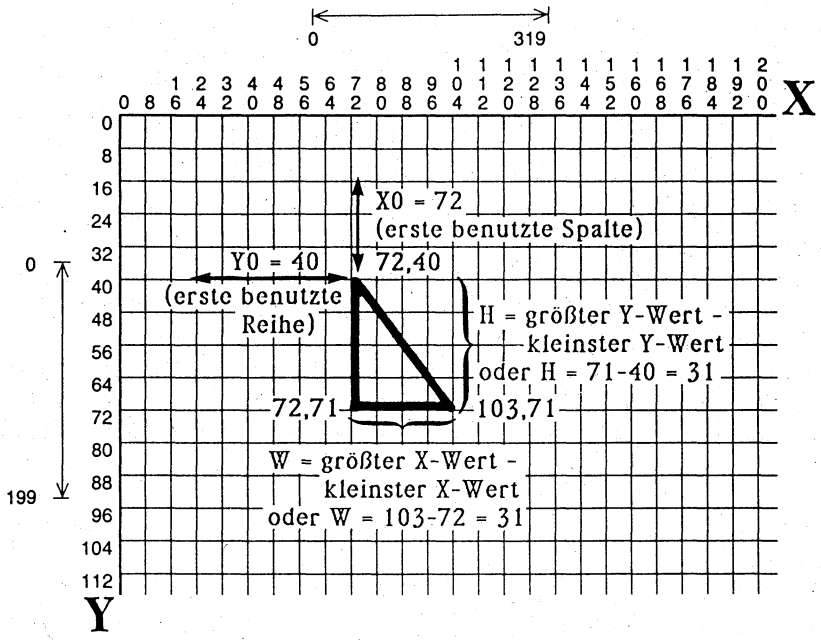
```
90 REM::::::::PAINT A SHAPE
91 PC = PC + ABS(PC = 0): FOR X = X0 TO
   X0 + W: FL$ = "F": PR = 0
92 FOR YC = Y0 TO Y0 + H: Y = YC: GOSUB 60
93 ON ABS((PEEK(BYTE) AND 2 ^ BIT) <> 0)
   GOTO 97: IF PR = 0 THEN 96
94 PR = 0: IF FL$ = "F" THEN Y1 = YC: FL$ =
   "T": GOTO 96
95 GOSUB 99: FL$ = "F"
96 NEXT YC: GOTO 98
97 PR = 1: NEXT YC: IF FL$ = "T" THEN GOSUB 99
98 NEXT X:  RETURN
99 FOR Y = Y1 TO YC - 1: ON ABS(RND(1) < PC)
   GOSUB 70: NEXT Y: RETURN
```

Um dieses neue Werkzeug zu benutzen, benötigen Sie den Befehl GOSUB 90. Vor GOSUB 90 benötigen Sie aber auch Zeilen der Hauptroutine, die folgendes beinhalten:

1. Die erste X-Spalte, in die Ihre Form fällt (X0 = ?).
2. Die erste Y-Reihe, in die Ihre Form fällt (Y0 = ?).
3. Die Breite Ihrer Form (W = ?). Beachten Sie: Die Breite beginnt bei 0. Wenn Ihre Form also 8 Pixelspalten breit ist, ist W = 7. Sie werden später noch sehen, daß eine bei 0 beginnende Breitenangabe leichter zu berechnen ist, als die tatsächliche Breite.
4. Die Höhe Ihrer Form (H = ?). Beachten Sie: Die Höhe beginnt bei 0. Wenn Ihre Form also 10 Pixelreihen hoch ist, ist H = 9. Auch hier werden Sie noch sehen, daß eine bei 0 beginnende Höhenangabe leichter zu berechnen ist, als die tatsächliche Höhe.

5. Der prozentuale Anteil der Pixels, die innerhalb der Form eingezeichnet werden sollen (PC = ?). Beachten Sie: Die Prozentzahl wird als Bruch angegeben. Setzen Sie also PC = .9, um willkürlich 90% der Form auszumalen; setzen Sie PC = .6, um willkürlich 60% der Form auszumalen; und setzen Sie PC = 1, um 100% der Form auszumalen. Alle nicht eingezeichneten Pixels haben die Hintergrundfarbe.

Schauen Sie sich die folgende Abbildung an, um zu sehen, wie dies bei Ihrem rechtwinkligen Dreieck funktioniert:



1. Die erste Spalte, die das Dreieck benutzt, ist die Spalte 72. Also X0 = 72.
2. Die erste Reihe, die das Dreieck benutzt, ist die Reihe 40. Also Y0 = 40.
3. Um die Breite zu bestimmen, substrahieren Sie die kleinste X-Koordinate, die für die Form angegeben ist, von der größten X-Koordinate. Für Ihr Dreieck bedeutet dies 103-72 oder W = 31.
4. Um die Höhe zu bestimmen, substrahieren Sie die kleinste Y-Koordinate, die für die Form angegeben ist, von der größten

Y-Koordinate. Für Ihr Dreieck bedeutet dies 71-40 oder H = 31.

5. Wir setzen PC = 1, so daß das Dreieck völlig ausgemalt wird.

Geben Sie die folgenden Programmzeilen ein, die die Umrisse Ihrer Form zeichnen:

```
4100 REM::::::::PAINT SHAPE TEST
4110 X0 = 72: Y0 = 40
4120 W = 31: H = 31
4130 PC = 1: GOSUB 90
```

Das ist alles. Bevor Sie das Programm starten, ändern Sie die Farbe für das Dreieck. Am Besten ändern Sie C bevor die Umrisse des Dreiecks eingezeichnet werden. Fügen Sie die Zeile 3000 C = 33 (Rot vor Weiß) hinzu. Mit dieser Änderung von C in Zeile 3000 können Sie ganz sicher sein, daß jedes eingezeichnete Pixel ab Zeile 3000 die neue Vordergrundfarbe erhält. Zum Schluß fügen Sie noch die Zeile 1130 GOTO 3000 ein, so daß die neue Farbe vom Computer verwendet wird. Starten Sie das Programm.

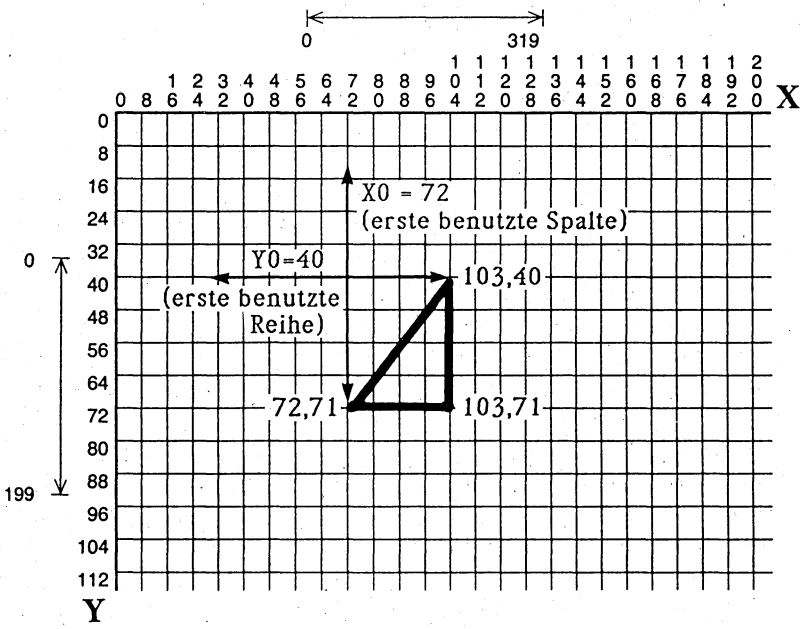
Der Bildschirm wird gelöscht und die Pixels erhalten die weiße Hintergrundfarbe (dies dauert einen Moment). Dann werden die Umrisse des Dreiecks in Rot eingezeichnet. Zum Schluß wird die Form vollständig ausgemalt. Beobachten Sie genau, wie das Dreieck gezeichnet wird. In der ersten benutzten Reihe (Y0 = 40) und der ersten benutzten Spalte (X0 = 72) beginnend, zeichnet der Computer die Spalte innerhalb der Umrisse hinunter. Wenn die Spalte fertig ist, wird die nächste (73) innerhalb der Umrisse von oben nach unten eingezeichnet. Dies wird solange fortgeführt, bis das Dreieck ganz ausgefüllt ist.

Wenn das Dreieck nicht korrekt eingezeichnet wird, brechen Sie das Programm ab und überprüfen Sie die neuen Programmzeilen. Das Problem liegt möglicherweise in der Unterroutine PAINT A SHAPE, achten Sie also besonders auf diese Zeilen.

Es ist hier besonders wichtig, zu verstehen, wie die Unterroutine arbeitet, im wesentlichen, weil sie nicht jede Art von Formen zeichnen kann. Lesen Sie den Rest dieses Kapitels sehr sorgfältig, sonst erleben Sie vielleicht unliebsame Überraschungen, wenn Sie später Ihre eigenen Formen zeichnen.

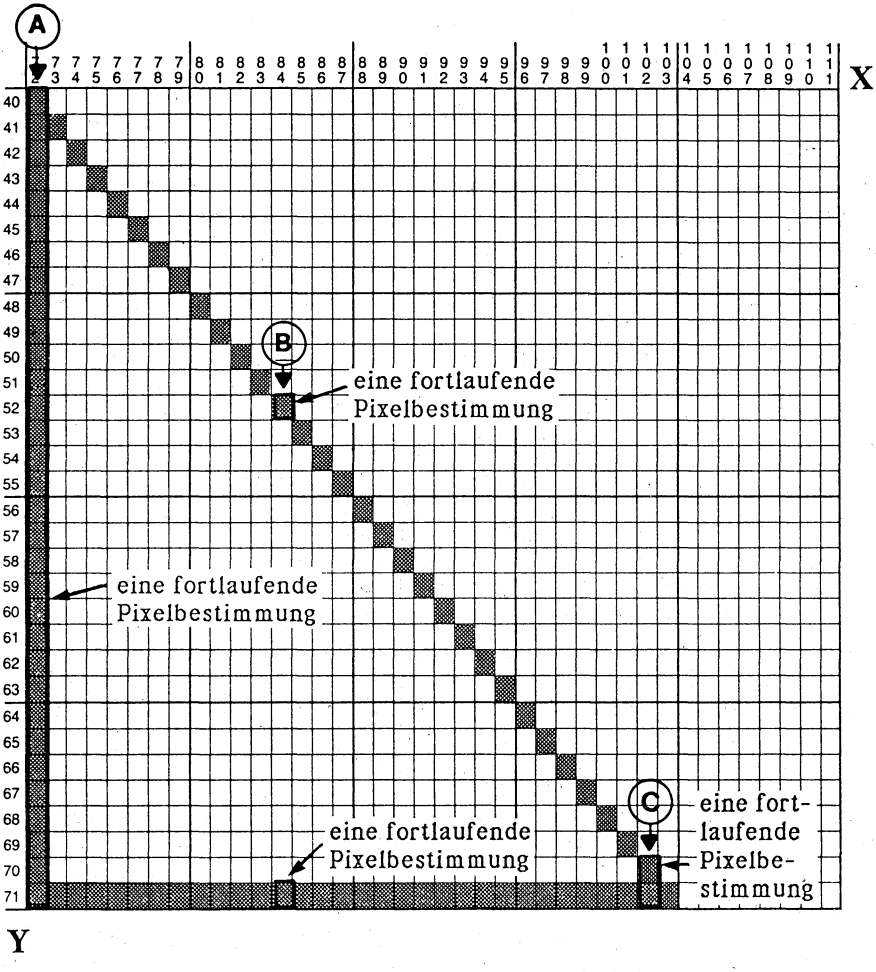
Um eine Form zu zeichnen, beginnt der Computer bei Spalte X0 und Reihe Y0 (entspricht Punkt X0,Y0), die in der Hauptroutine angegeben sind. Diese Pixelstelle gibt dem Computer die erste Spalte und Reihe an, die für die Umrisse der Form benutzt werden. Beachten Sie aber, daß dieses Pixel nicht unbedingt ein Teil der

Umrisse sein muß. Angenommen, Ihr rechtwinkliges Dreieck wäre genau spiegelbildlich zu dem schon gezeichneten, wie in der Abbildung unten, dann würde das spiegelbildliche Dreieck immer noch in derselben ersten Spalte (X0 = 72) und derselben ersten Reihe (Y0 = 40) erscheinen, aber der Punkt 72,40 läge nicht mehr auf den Umrißlinien des Dreiecks. Der Computer benutzt den Punkt X0,Y0 nur als Startpunkt.



Die Form wird Spalte für Spalte gezeichnet, beginnend bei Punkt X0,Y0, die Spalte hinunter, bis die angegebene Länge (H = 31) erreicht ist, und spaltenweise weiter, bis die angegebene Breite (W = 31) erreicht ist. Das Ausmalen erfolgt aber nur in den Spalten, die zwei, jeweils fortlaufende Bestimmungen von Umrißpixels enthalten.

"Umrißpixels" sind die eingezeichneten Vordergrundpixels, die die Umrisse der Form bilden, wodurch jede eingezeichnete Linie der Form aus diesen Umrißpixels besteht. "Fortlaufende Bestimmung" bedeutet, daß ein oder mehrere Umrißpixels direkt untereinander in einer Spalte innerhalb Ihrer Form erscheinen. Sehen Sie sich die Vergrößerung Ihrer Dreiecksumrisse an, damit Sie verstehen, was wir meinen:

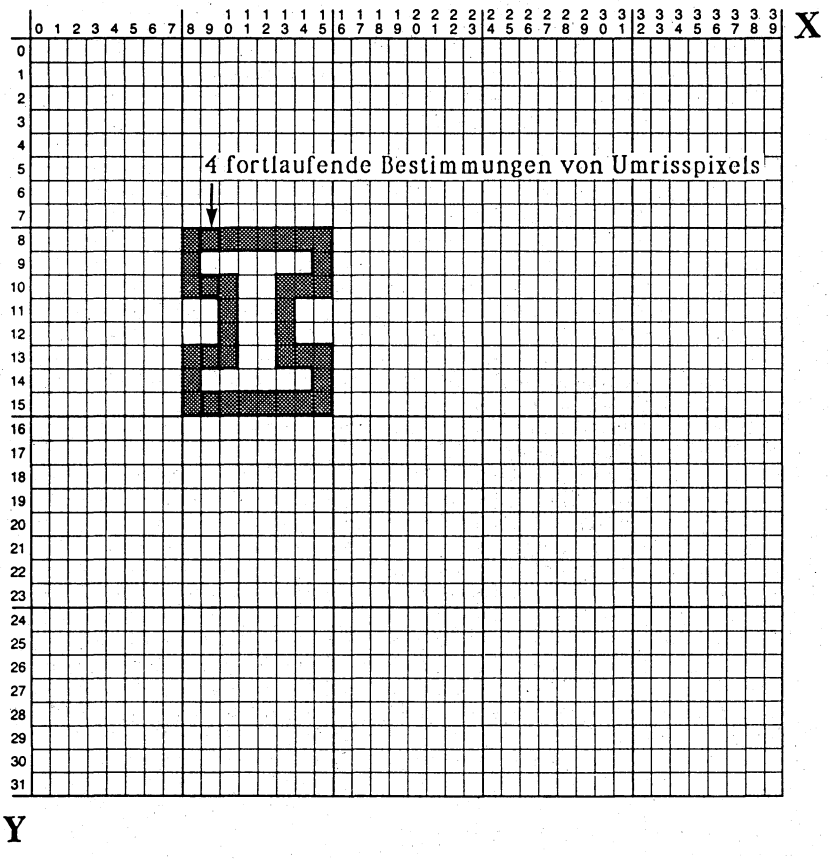


Die erste Spalte in Ihrem Dreieck (A) besteht aus nur einer fortfaufenden Bestimmung von Umrißpixels und kann so von der Unterroutine übersprungen werden. (Da Spalten von der Unterroutine übersprungen werden können, ist es wichtig, daß Sie die Umrisse in derselben Farbe einzeichnen, mit der Sie die Form später auch ausmalen wollen.)

Spalte B in der Zeichnung beinhaltet zwei fortlaufende Bestimmungen von Umrißpixels. (Noch einmal: eine "fortlaufende Bestimmung" kann aus nur einem Umrißpixel bestehen.) In diesem Fall zeichnet die Unterroutine jedes Pixel zwischen den beiden Bestimmungen ein.

Spalte C enthält nur eine fortlaufende Bestimmung von Umrißpixels und wird von der Unterroutine übersprungen.

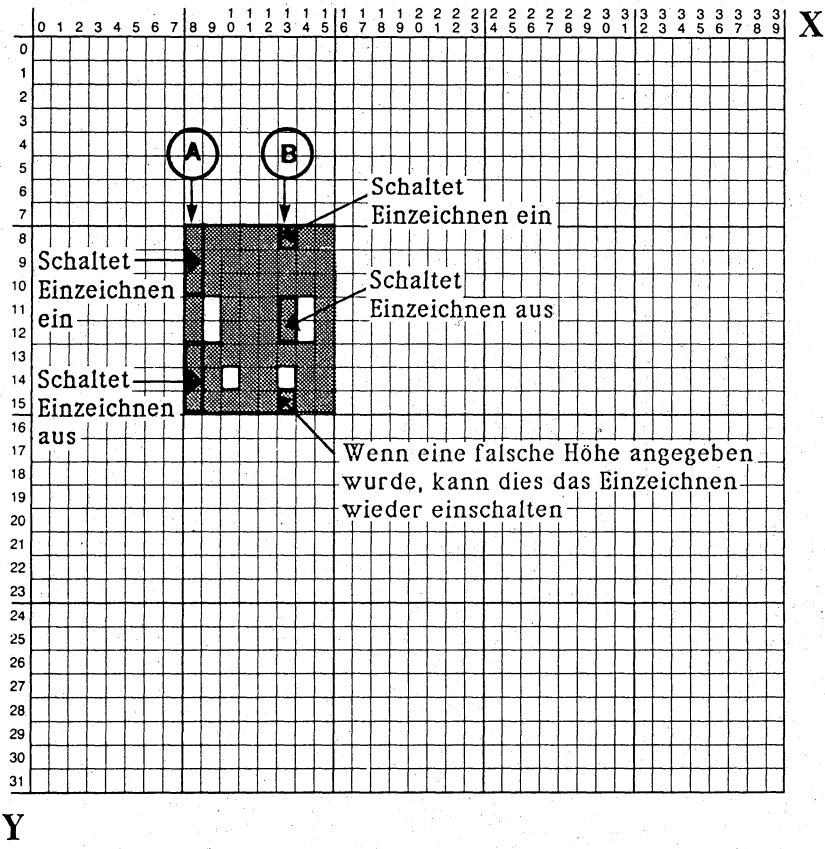
Es kann aber auch vorkommen, daß eine Spalte mehr als zwei fortlaufende Bestimmungen von Umrißpixels enthält. Sehen Sie sich die Form in der folgenden Abbildung an:



In dem obigen Diagramm haben wir eine Spalte markiert, die vier fortlaufende Bestimmungen von Umrißpixels enthält. Die Unterroutine benutzt jede Bestimmung als einen EIN/AUS-Schalter zum Einzeichnen. Wenn die erste Bestimmung überschritten wird, beginnt der Computer einzuzeichnen. Wird die zweite Bestimmung überschritten, unterbricht der Computer das Einzeichnen. Wird die dritte Bestimmung überschritten, wird wieder eingezeichnet, bei der vierten Bestimmung wieder unterbrochen. Wenn es keine vierte Bestimmung gäbe, würde das Einzeichnen in der untersten Reihe der Form, angegeben durch H = ? in der Hauptroutine, beendet.

Dieser EIN/AUS-Schalter wird in jeder Spalte benutzt, auch in den Spalten, die zwei fortlaufende Bestimmungen von Umrißpixels beinhalten. Der Computer beginnt das Einzeichnen nach der ersten Bestimmung und unterbricht, wenn er die zweite erreicht. Dies ist der wichtigste und vielleicht schwer zu verstehende Bestandteil des neuen Werkzeugs. Jede fortlaufende Bestimmung ist ein Schalter zum Einzeichnen.

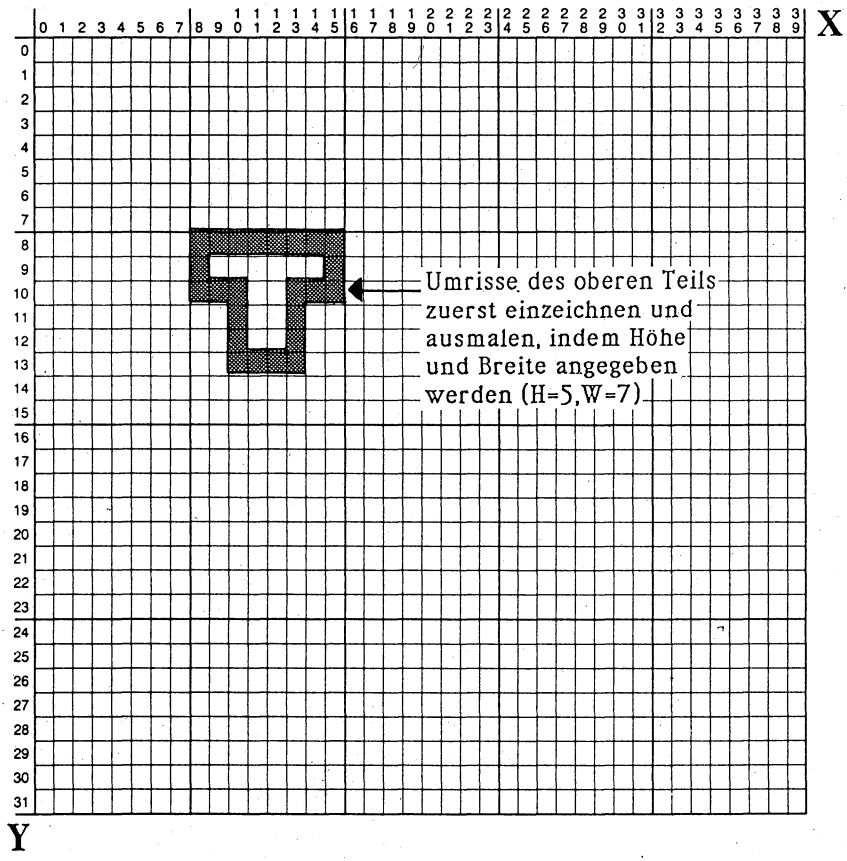
Wenn wir die oben abgebildete Form einzeichnen und ausmalen wollten, würde folgendes dabei entstehen:

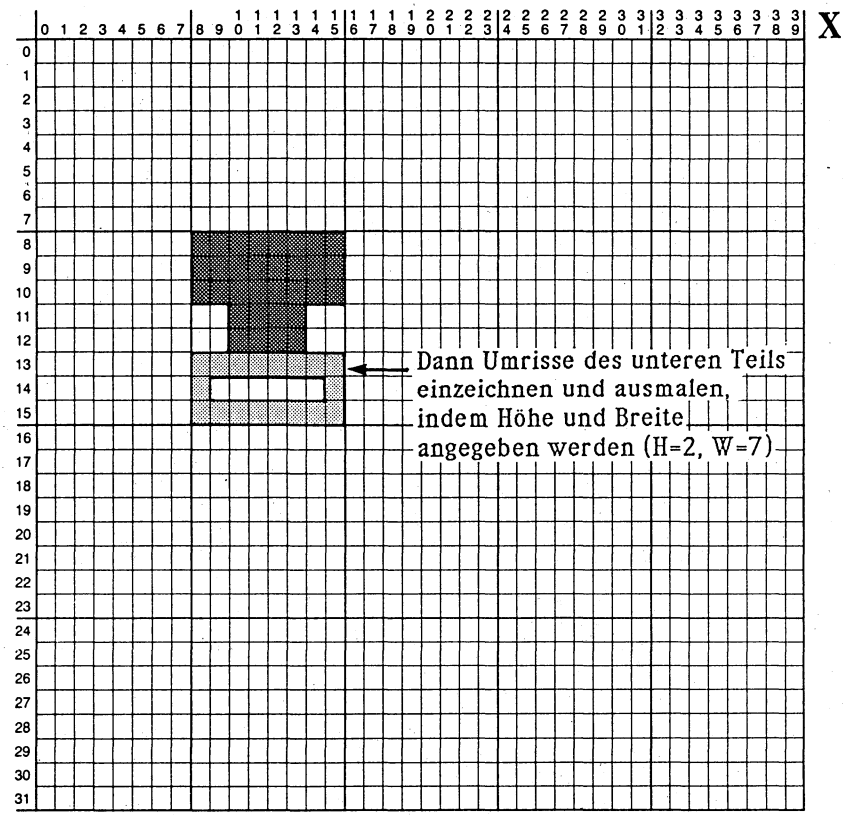


Spalte A enthält zwei Bestimmungen von Umrißpixels. Die erste würde das Einzeichnen einschalten, die zweite ausschalten. Das Ergebnis wären unerwünscht eingezeichnete Pixels. Dies würde auch in der letzten Spalte der Form auftreten.

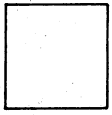
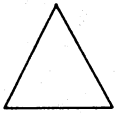
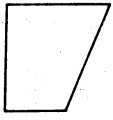
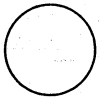
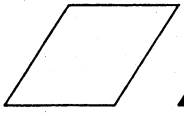
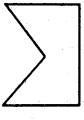
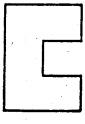
Spalte B beinhaltet drei Bestimmungen von Umrißpixels. Die erste würde das Einzeichnen, wie gewünscht, einschalten, die zweite würde es hingegen wieder ausschalten, so daß die folgenden Pixels nicht eingezeichnet würden. Wenn die dritte Bestimmung überschritten würde, wäre das Ende der Spalte erreicht. Das Einzeichnen würde unterbrochen, unabhängig vom Vorhandensein oder Fehlen eines Umrißpixels. So würde die Form eingezeichnet und ausgemalt und nur die Höhe würde das Einzeichnen beenden. Wenn eine falsche Höhe in der Hauptroutine angegeben wäre, würde das Einzeichnen nach der dritten Bestimmung wieder beginnen.

Lassen Sie sich nicht entmutigen, wenn Ihnen Formen einfallen, die mit Ihrem neuen Werkzeug nicht gezeichnet werden können. Jede Form kann geteilt und in Abschnitten mit dem neuen Werkzeug ausgemalt werden. Die Umrisse unserer Form "I" sind leicht zu teilen und auszumalen, wie Sie in den folgenden Abbildungen sehen:

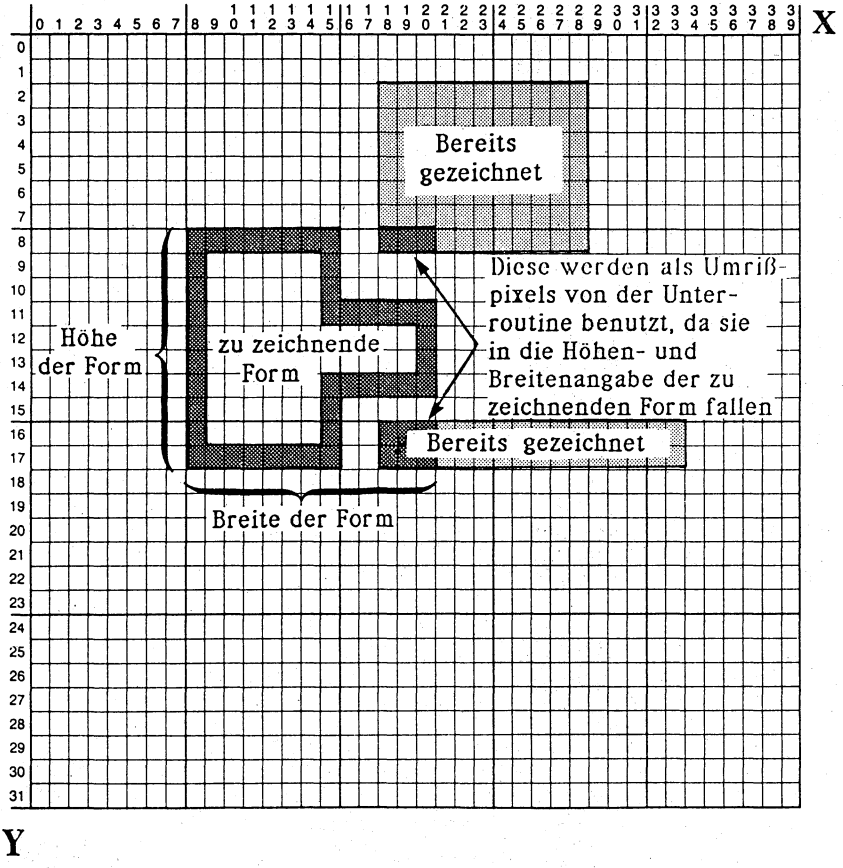



X
Y
Dann Umrisse des unteren Teils einzeichnen und ausmalen, indem Höhe und Breite angegeben werden (H=2, W=7)

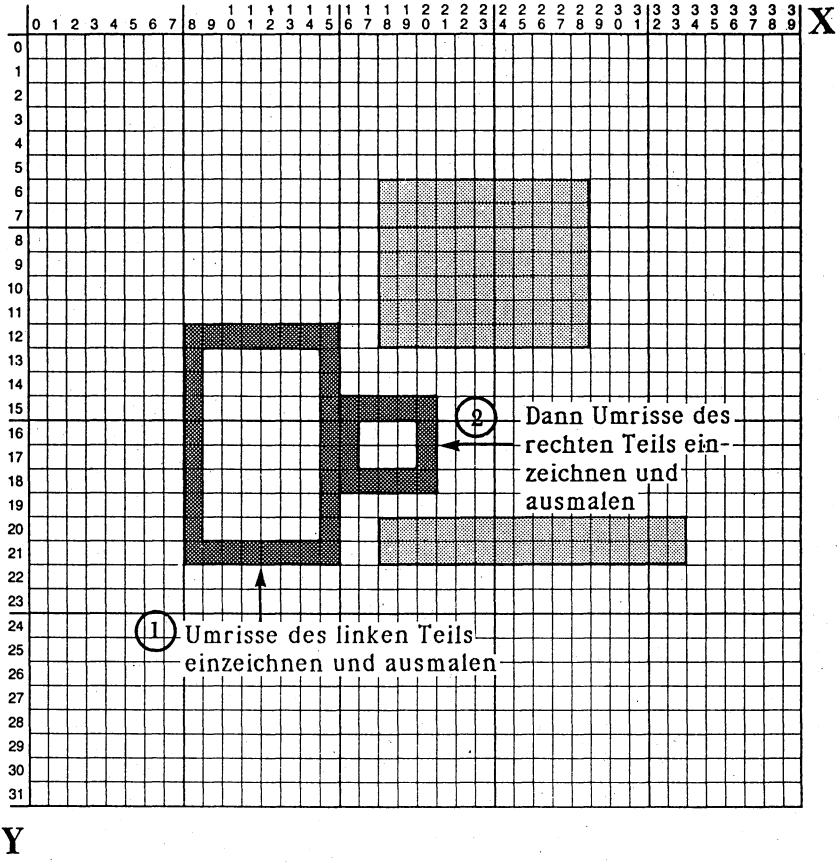
Unten sehen Sie häufig benutzte Formen, die "so wie sie sind" mit der Unterroutine PAINT A SHAPE ausgemalt werden können, erweitert um zwei Formen, die geteilt werden müssen, um sie mit der Routine PAINT A SHAPE auszumalen.

Umrisse der Form	Ausgemalte Form	Andere Möglichkeit des Ausmalens
		nicht notwendig
		nicht notwendig
		nicht notwendig
		nicht notwendig
		nicht notwendig
		obere Hälfte einzeichnen und ausmalen, dann untere Hälfte einzeichnen und ausmalen
		obere Hälfte einzeichnen und ausmalen, dann untere Hälfte einzeichnen und ausmalen

Beim Ausmalen von Formen müssen Sie immer daran denken, daß die Unterroutine in Spalte Y0 beginnt, wenn sie die Spalten einzeichnet. Sie geht die Spalte hinunter und beginnt unmittelbar unter der ersten fortlaufenden Bestimmung von Umrißpixels, zu der sie gelangt, einzuzeichnen. Schauen Sie sich das untenstehende Diagramm an: Sie sehen eine Form, die gezeichnet werden soll, und zwei schon gezeichnete Formen.



Die vorher gezeichneten Formen liegen innerhalb der Spalten der zu zeichnenden Form. Ihre Vordergrundpixels werden von der Unterroutine PAINT A SHAPE als Umrißpixels betrachtet und als solche schalten sie das Einzeichnen in jeder Spalte, in der sie erscheinen, ein und aus. Um dieses Problem zu bewältigen, teilen Sie die zu zeichnende Form wieder auf:



Suchen Sie in Ihren skizzierten Bildern zur Kontrolle nach:

1. 3 Farben in einem Block;
2. 2 Vordergrundfarben in einem Block; und
3. fortlaufenden Bestimmungen von Umrißpixels die das Einzeichnen ein- oder ausschalten, wenn eine Form ausgemalt wird.

Jede Form kann mit Ihrem neuen Werkzeug in ihren Umrissen eingezeichnet und dann ausgemalt werden. Sehen Sie sich Ihre skizzierten Bilder sehr sorgfältig an. Sie können vorher die Formen bestimmen, die aufgeteilt werden müssen, um sie zu zeichnen. Wenn Sie das Programm eingeben, vergewissern Sie sich, daß Sie die Farben für die Umrisse benutzen, mit denen Sie die Formen nachher auch ausmalen wollen.

Wenn Sie Zeit dazu haben, zeichnen Sie einige selbstentworfene Formen ein und malen sie aus. Der Mehraufwand lohnt sich. Wenn

Sie Ihre eigenen Bilder entwerfen wollen, ist es wichtig, daß Sie verstehen, wie das Werkzeug PAINT A SHAPE die Vordergrundpixels in jeder Spalte jedweder Form benutzt.

Die Umrisse des Landes und der Wellen

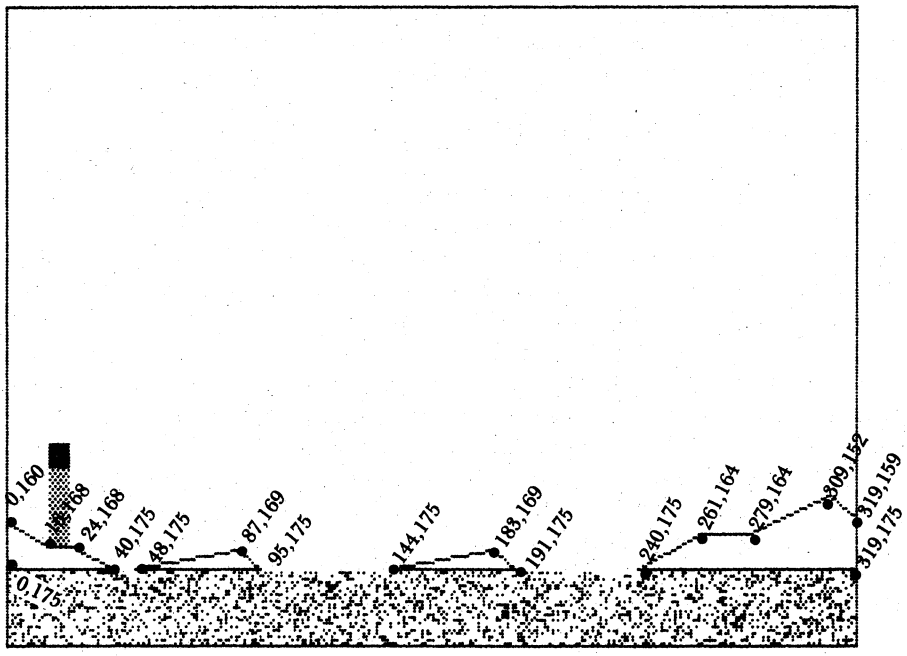
Um die Umrisse von Landstücken und Wellen einzuzeichnen, müssen Sie zuerst die Hauptroutine aus Kapitel 3 neu eingeben. (Wenn Sie das Programm aus Kapitel 3 laden würden, wären die beiden neuen Unterroutinen verloren.)

Geben Sie RUN 10 RETURN ein und warten Sie, bis die Zeilen der Übungshauptroutine gelöscht sind. Dann geben Sie die Hauptroutine aus Kapitel 3 wie folgt ein:

```
1100 GOSUB 20          : REM GRAPHICS
1110 C = 14: GOSUB 40: REM COLORS
1120 GOSUB 50          : REM PAINT BKGROUND
1200 REM:::::::::LIGHTHOUSE
1210 POKE 18090,0: POKE 18130,17
1220 POKE 18170,17: POKE 18210,17
1300 REM:::::::::WATER
1310 FOR Y = 176 TO 199
1320 FOR X = 0 TO 319
1330 IF RND(1) < .3 THEN GOSUB 70
1340 NEXT X: NEXT Y
5000 GET A$
5010 IF A$ = " " THEN 6000
5020 GOTO 5000
6000 GOSUB 30
6010 END
```

Starten Sie das Programm zur Überprüfung der Eingabe. Das Einzeichnen des Wassers dauert eine Weile, Sie können also eine Pause machen. Nach 15-20 Minuten müßten Himmel, Wasser und der Leuchtturm eingezeichnet sein. Betätigen Sie dann die LEERTASTE, um das Programm-Listing wieder zu erhalten.

Unten sehen Sie eine Skizze der Umrisse für das Land und die Wellen. Die X,Y-Koordinaten für jeden Pixelpunkt, der den Anfang oder das Ende einer Linie bildet, sind darin angegeben. Diese Koordinaten werden in dem neuen Programm benutzt, um die Umrisse einzuzeichnen.



Geben Sie die folgenden Zeilen ein, die die Umrisse des linken und rechten Landteils zeichnen. Vergleichen Sie die Programmzeilen mit den X,Y-Koordinaten in der obigen Skizze.

```
1400 REM:::::::::LEFT LAND
1410 C = 94: REM COLOR = GREEN ON BLUE
1420 X1 = 0: Y1 = 160
1430 X2 = 16: Y2 = 168: GOSUB 80
1440 X1 = 24: Y1 = 168: GOSUB 80
1450 X2 = 40: Y2 = 175: GOSUB 80
1460 X1 = 0: Y1 = 175: GOSUB 80
1470 X2 = 0: Y2 = 160: GOSUB 80
1500 REM:::::::::RIGHT LAND
1510 X1 = 240: Y1 = 175
1520 X2 = 261: Y2 = 164: GOSUB 80
1530 X1 = 279: Y1 = 164: GOSUB 80
1540 X2 = 309: Y2 = 152: GOSUB 80
1550 X1 = 319: Y1 = 159: GOSUB 80
1560 X2 = 319: Y2 = 175: GOSUB 80
1570 X1 = 240: Y1 = 175: GOSUB 80
```

Überprüfen Sie jede Zeile und korrigieren Sie jeden Fehler, den Sie finden. Wenn Sie fertig sind, bewegen Sie den Cursor zu einer leeren Zeile unter Ihrem Programm und geben sorgfältig ein:

GOSUB 20: RUN 1400 RETURN

Das ist eine neue Methode, Teile Ihres Programms zu umgehen. GOSUB 20 schaltet den hochauflösenden Bildschirm ein. RUN 1400 befiehlt dem Computer, das Programm bei Zeile 1400 zu starten. Wenn Sie Befehle ohne Zeilennummern eingeben, werden sie ausgeführt, aber nicht in das Programm eingefügt.

Vergleichen Sie das, was auf Ihrem Bildschirm geschieht, mit dem letzten Diagramm. Die Umrisse des Landes müßten ähnlich dem Diagramm in grünen Vordergrundpixels eingezeichnet werden.

Sind die Umrisse eingezeichnet, betätigen Sie die LEERTASTE. Wenn Probleme bei der Ausführung des Programms aufgetaucht sind, überprüfen Sie jede Zeile sorgfältig. Sehen Sie sich alle X- und Y-Koordinaten an. Häufig werden X und Y vertauscht oder es wird X1 (Y1) anstelle von X2 (Y2) eingegeben. Korrigieren Sie alle Schreibfehler. Für einen weiteren Versuch müssen Sie das gesamte Programm noch einmal laufen lassen. Da Zeile 1120 alle Vordergrundpixels zu Hintergrundpixels ändert, muß der ganze Vorgang noch einmal durchgeführt werden, wenn Sie die Umrisse falsch eingezeichnet haben.

Fahren wir fort mit den Umrissen für die linke und rechte Welle:

```
1600 REM:::::::::LEFT WAVE
1610 C = 14: REM COLOR = BLACK ON BLUE
1620 X1 = 48: Y1 = 175
1630 X2 = 87: Y2 = 169: GOSUB 80
1640 X1 = 95: Y1 = 175: GOSUB 80
1650 X2 = 48: Y2 = 175: GOSUB 80
1700 REM:::::::::RIGHT WAVE
1710 X1 = 144: Y1 = 175
1720 X2 = 183: Y2 = 169: GOSUB 80
1730 X1 = 191: Y1 = 175: GOSUB 80
1740 X2 = 144: Y2 = 175: GOSUB 80
```

Um diese Umrisse in Ihr Bild einzufügen, bewegen Sie den Cursor auf eine leere Zeile und geben Sie ein:

GOSUB 20: RUN 1600 RETURN

Sehen Sie sich noch einmal das Diagramm mit den Umrissen der Formen an. Vergewissern Sie sich, daß das Programm die Umrisse korrekt einzeichnet. Wenn beide Wellen fertig sind, betätigen Sie die LEERTASTE.

Sehen Sie sich die Zeilen 1700 bis 1740 an. Sie erinnern sich, daß die Umrisse einer Form aus mehreren verbundenen Linien bestehen. Um Linien einzuzeichnen, sind die X1,Y1- und die X2,Y2-Koordinaten notwendig, die mit GOSUB 80 zu einer Linie verbunden werden. Um Linien einzuzeichnen, benutzt der Computer die zuletzt im Programm angegebenen X1,Y1- und X2,Y2-Koordinaten, so daß Sie diese wieder benutzen können, ohne sie neu einzugeben.

Werkzeug 80::::::::::Einzeichnen einer Linie

```
80 REM::::::::::PLOT A LINE
81 DX = X2 - X1: DY = Y2 - Y1
82 L = ABS(DX): IF ABS(DY) > L THEN
   L = ABS(DY)
83 IF L > 0 THEN XI = DX/L: YI = DY/L
84 X = X1 + .5: Y = Y1 + .5
85 FOR I = 0 TO L
86 GOSUB 70: REM PLOT A POINT
87 X = X + XI: Y = Y + YI
88 NEXT I
89 RETURN
```

Zweck: *Diese Routine zeichnet eine gerade Linie in der Vordergrundfarbe, die durch den aktuellen Wert von C angegeben ist und zwar von den Koordinaten X1,Y1 zu X2,Y2.*

Anwendung: *Um eine Linie einzuzeichnen, benötigen Sie eine oder mehrere Zeilen in der Hauptroutine, um den Anfangspunkt (X1,Y1) und den Endpunkt (X2,Y2) der einzuzeichnenden Linie anzugeben, wozu der Befehl GOSUB 80 notwendig ist. Ein Beispiel:*

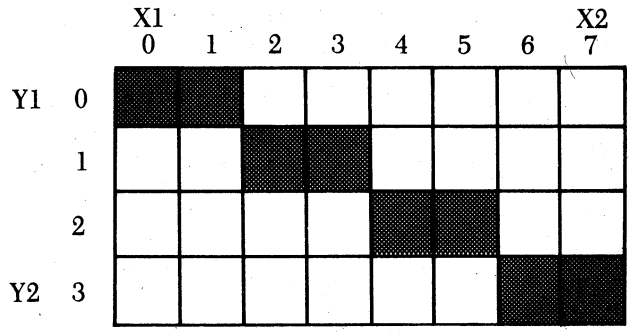
```
1110 X1 = erste X-Koordinate: Y1 = erste
Y-Koordinate
1120 X2 = letzte X-Koordinate: Y2 = letzte
Y-Koordinate
1130 GOSUB 80
```

Wenn Sie mehrere Linien als Umrisse einer Form einzeichnen, sollte als Wert von C der Farbcode eingesetzt werden, der später

auch benutzt wird, um die Form auszumalen. Dies muß vor dem Befehl GOSUB 80 geschehen. Ein Beispiel:

```
1100 C = 14
```

Technische Beschreibung: *Um zu verstehen, wie eine Linie gezeichnet wird, stellen Sie sich einen Punkt vor, der von Punkt X1,Y1 zu Punkt X2,Y2 wandert und jedes Pixel auf seinem Weg zur Vordergrundfarbe ändert. Um zu Punkt X2,Y2 zu gelangen, muß der wandernde Punkt das direkteste Bewegungsmuster und die Anzahl der Wiederholungen dieses Musters errechnen, um sein Ziel zu erreichen. Wir benutzen diese Begriffe "Muster" und "Wiederholungen", um die Unterroutine zu erklären.*



Das Muster in obigem Beispiel ist folgendes: <einzeichnen>; 1 Schritt rechts; 1/2 Schritt nach unten. Es erfolgen 8 Wiederholungen oder eine achtmalige Ausführung dieses Musters.

Zwei weitere hilfreiche Begriffe zur Erklärung dieser Unterroutine lauten "Hauptachse" und "Nebenachse". In obigem Beispiel ist der Weg von X1 zu X2 weiter als der von Y1 zu Y2, also ist X die "Hauptachse" und Y die "Nebenachse". Wenn die Entfernung von Y1 zu Y2 größer wäre, als von X1 zu X2, wäre Y die Hauptachse.

Der wandernde Punkt folgt immer einem Standardmuster:

1. *<Einzeichnen>*
2. *1 Schritt entlang der Hauptachse*
3. *1 Schritt oder weniger entlang der Nebenachse*

Um eine Linie einzuzeichnen, muß der Punkt:

1. *Entscheiden, welche die Haupt- und welche die Nebenachse ist; entscheiden, wieviele Musterwiederholungen er durchzuführen hat; entscheiden, wie weit er in jeder Wiederholung entlang der Nebenachse für jeden ganzen Schritt entlang der Hauptachse wandern muß. (Dies ist die einzige Variable, neben der Haupt- und der Nebenachse, die das Muster beeinflußt.)*

```
81 DX = X2 - X1: DY = Y2 - Y1
```

Diese Zeile berechnet die Entfernung von X1 zu X2(DX) und von Y1 zu Y2(DY). Die größere dieser beiden Zahlen bestimmt die Hauptachse, die kleinere die Nebenachse.

```
82 L = ABS(DX): IF ABS(DY) > L THEN L = ABS(DY)
```

Diese Zeile berechnet die Anzahl der Musterwiederholungen, die notwendig sind, um die Linie zu zeichnen. Da der wandernde Punkt bei jeder Wiederholung einen Schritt entlang der Hauptachse ausführt, kann deren Länge als die Anzahl der Wiederholungen benutzt werden. Die Anzahl der Wiederholungen wird in der Variablen L gespeichert. Zeile 82 beginnt mit der Annahme, daß X die Hauptachse ist, und deshalb ABS(DX) gleich "Anzahl der Wiederholungen" ist. Dann überprüft sie, ob die Annahme, daß X die Hauptachse ist, richtig ist. Wenn sich herausstellt, daß eigentlich Y die Hauptachse ist, setzt sie die Anzahl der Wiederholungen mit ABS(DY) gleich.

```
83 IF L > 0 THEN XI = DX/L: YI = DY/L
```

Diese Zeile berechnet das Muster, das der wandernde Punkt wiederholt, um sein Ziel zu erreichen. Als erstes prüft die Zeile, ob die Anzahl der Wiederholungen größer als (<) Null ist. Wenn nicht, wandert der Punkt überhaupt nicht. Er zeichnet nur den Punkt an der Stelle X1,Y1 und bleibt dort. Andernfalls berechnet die Zeile, wie weit der Punkt bei jeder Wiederholung entlang der X-Achse (XI) und entlang der Y-Achse wandern muß. Da L entweder gleich ABS(DX) oder ABS(DY) ist, ist eine dieser Zahlen (XI oder YI) 1 (-1, wenn er sich zur Achse bewegt). Die andere Zahl ist kleiner als oder gleich (> =) 1. Indem diese Zahlen (an Wert zunehmend) nach jedem Einzeichnen eines Punktes zur aktuellen Position des wandernden Punktes addiert werden, bewegt sich der Punkt im angestrebten Muster.

```
84 X = XI + .5: Y = YI + .5
```

Zeile 84 plaziert den wandernden Punkt an den Anfangspunkt der einzuzeichnenden Linie. Die Addition von .5 zu jeder Koordinate liefert eine genauere Startposition. Da die Routine zum Einzeichnen von Punkten nur die ganzen Zahlen unter den X,Y-Koordinaten benutzt, würde eine X-Koordinate von 4.99 zu 4. Durch die Addition von .5 zur X-Koordinate wird diese zu 5.49, bzw. bei der Verarbeitung der Koordinate durch die Routine zu 5, was offensichtlich viel näher an dem ursprünglichen Wert 4.99 liegt. Dieser Vorgang wird "Runden" genannt.

```
85 FOR I = 0 TO L
86 GOSUB 70: REM PLOT A POINT
87 X = X + XI: Y = Y + YI
88 NEXT I
```

Diese Schleife erledigt die ganze Arbeit. Zeile 85 bildet eine Schleife für jede Musterwiederholung, Zeile 86 zeichnet den Punkt ein, auf dem der wandernde Punkt sich gerade befindet, Zeile 87 addiert den wachsenden Wert, wodurch der wandernde Punkt sich an die nächste Position bewegt und Zeile 88 schließlich kehrt für eine weitere Wiederholung an den Anfang der Schleife zurück.

```
89 RETURN
```

Diese Zeile bewirkt, daß der Computer zur Hauptroutine zurückkehrt, nachdem die Linie eingezeichnet worden ist.

Ausmalen der Landstücke und der Wellen

Wenn Sie die Umrisse eingezeichnet haben, können Sie die Flächen innerhalb der Umrisse ausmalen. Für die Wellen geben Sie die folgenden Programmzeilen ein:

```
1800 REM:::::::::PAINT LEFT WAVE
1810 C = 14: REM COLOR = BLACK ON BLUE
1820 X0 = 48: Y0 = 169
1830 W = 47: H = 6
1840 PC = .3: GOSUB 90
1900 REM:::::::::PAINT RIGHT WAVE
1910 X0 = 144: Y0 = 169
```

```
1920 W = 47: H = 6
1930 PC = .3: GOSUB 90
```

Das Ausmalen erfolgt innerhalb der Ausmaße (Höhe/Breite) der Form, beginnend bei Punkt X0,Y0, der im Programm angegeben wird. Die Höhe und Breite einer Form wird durch eine Skala bestimmt, die bei 0 beginnt. Da die linke Welle 48 Spalten breit und 7 Reihen hoch ist, wird für W 47 und für H 6 angegeben. PC steht für den prozentualen Anteil der Pixels, die in der Form eingezeichnet werden. Für das Wasser wurden willkürlich 30% der Pixels eingezeichnet, für die Wellen gilt dasselbe (PC = .3) Bewegen Sie den Cursor auf eine leere Zeile und geben Sie ein:

```
GOSUB 20: RUN 1800 RETURN
```

Schauen Sie sich an, wie jede Welle innerhalb der Umrisse mit zufällig bestimmten Pixels ausgemalt wird. Das Einzeichnen der Pixels erfolgt spaltenweise, zwischen den fortlaufenden Bestimmungen der Umrißpixels. Wenn das untere Ende der Form erreicht ist, wird in der nächsten Spalte eingezeichnet, solange, bis die letzte Spalte der Form ausgefüllt ist.

Wenn die Wellen ausgemalt sind, betätigen Sie die LEERTASTE. Dann geben Sie nachfolgende Programmzeilen, die das Kolorieren der Landumrisse besorgen, ein.

```
2000 REM:::::::::PAINT LEFT LAND
2010 C = 94: REM COLOR = GREEN ON BLUE
2020 X0 = 0: Y0 = 160
2030 W = 40: H = 15
2040 PC = 1: GOSUB 90
2100 REM:::::::::PAINT RIGHT LAND
2110 X0 = 237: Y0 = 152
2120 W = 82: H = 23
2130 PC = 1: GOSUB 90
```

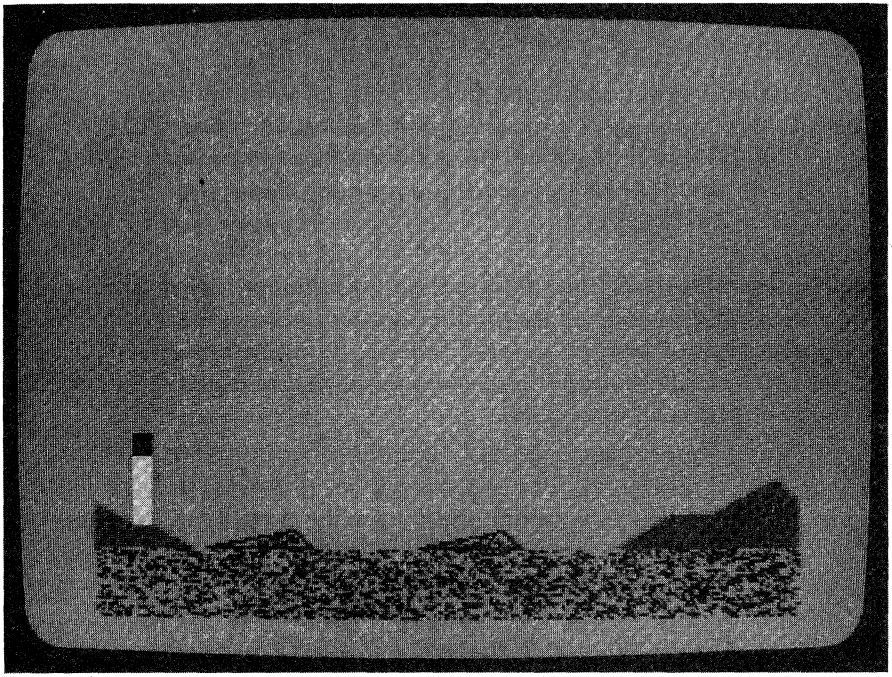
Für PC wird nun 1 eingesetzt, so daß die Bereiche des Landes als einheitliche Fläche völlig ausgefüllt werden. Geben Sie ein:

```
GOSUB 20: RUN 2000 RETURN
```

Sehen Sie sich an, was auf Ihrem Bildschirm passiert. Da jeder Block eingezeichnet wird, werden die Vordergrundpixels, die in dem Block schon vorhanden sind, zu Grün geändert. Auf diese Weise wird die senkrechte Umrißlinie des Landes ganz links zu Grün geändert.

Die Unterroutine selbst ignoriert diese Spalte eigentlich, da sie nur eine fortlaufende Bestimmung von Umrißpixels enthält. Nach einer kurzen Pause wird das Landstück auf der rechten Seite ausgemalt.

Vergleichen Sie Ihren Bildschirm mit der folgenden Abbildung, Wenn Sie nicht genau wissen, warum die neuen Zeilen der Hauptroutine diesen Teil Ihres Bildes erstellt haben, lesen Sie noch einmal die Abschnitte über das Einzeichnen von Linien und/oder über das Ausmalen von Formen.



Werkzeug 90:::::::::Ausmalen einer Form

```
90 REM:::::::::PAINT A SHAPE
91 PC = PC + ABS(PC = 0): FOR X = X0 TO
   X0 + W: FL$ = "F": PR = 0
92 FOR YC = Y0 TO Y0 + H: Y = YC: GOSUB 60
93 ON ABS((PEEK(BYTE) AND 2 ^ BIT) <> 0)
   GOTO 97: IF PR = 0 THEN 96
94 PR = 0: IF FL$ = "F" THEN Y1 = YC: FL$ =
   "T": GOTO 96
```

```
95 GOSUB 99: FL$ = "F"
96 NEXT YC: GOTO 98
97 PR = 1: NEXT YC: IF FL$ = "T" THEN GOSUB 99
98 NEXT X:  RETURN
99 FOR Y = Y1 TO YC - 1: ON ABS(RND(1) < PC)
   GOSUB 70: NEXT Y: RETURN
```

Zweck: *Diese Unterroutine füllt die meisten Formen mit der Vordergrundfarbe aus, die durch den aktuellen Wert von C angegeben ist.*

Anwendung: *Um dieses Werkzeug zu benutzen, benötigen Sie Zeilen der Hauptroutine, die*

- *die erste Spalte (X0 = ?) angeben, in die die Form fällt*
- *die erste Reihe (Y0 = ?) angeben, in die die Form fällt*
- *die Höhe (H = ?) und die Breite (W = ?) der Form mit einer bei 0 beginnenden Skala angeben. Zusätzlich müssen Sie den prozentualen Anteil der Pixels (PC = ?) angeben, die innerhalb der Form eingezeichnet werden sollen. Dies geschieht folgendermaßen:*

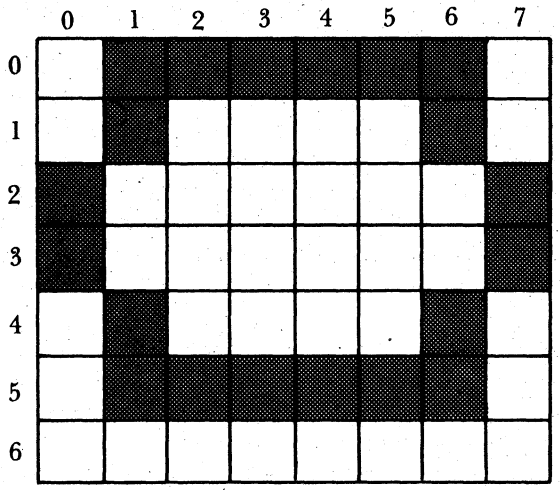
1100 X0 = X-Koordinate, die am weitesten links liegt: Y0 = Y- Koordinate, die am weitesten oben liegt
1110 W = größter X-Wert - kleinster X-Wert: H = größter Y-Wert - kleinster Y-Wert
1120 PC = prozentualer Anteil der einzuzeichnenden Pixels, ausgedrückt in einem Bruch: C = Vorder-/Hintergrund-Farbcode für die Form
1130 GOSUB 90

Technische Beschreibung: *Diese Unterroutine durchsucht alle Spalten in einem bestimmten Bereich nach solchen, die zwei Umrißpixels enthalten. Das obere Umrißpixel der Spalte ist der "Anfangspunkt", das untere Umrißpixel der "Endpunkt". Wenn diese beiden Punkte gefunden sind, zeichnet Zeile 99 die Pixels dazwischen ein.*

Der Befehl PC = PC+ABS(PC = 0) in Zeile 91 prüft, ob PC auf Null gesetzt ist. Wenn ja, wird PC automatisch auf 1 gesetzt, andernfalls ändert sich nichts. Dieser Befehl wurde für den Fall eingefügt, daß die Variable PC nicht bestimmt ist, bevor die Unterroutine aufgerufen wird. Wenn PC (prozentualer Anteil der Pixels, die eingezeichnet werden) überhaupt nicht bestimmt würde, wäre PC gleich Null und 0% der Pixels innerhalb der Form würden

eingezeichnet, was den Eindruck eines scheinbaren Ausfalls der Unterroutine zur Folge hätte. Der obige Befehl in der Unterroutine erleichtert die Lokalisierung eines Fehlers, wenn PC nicht richtig bestimmt ist.

Die Variable FL$ verfolgt, ob der Anfangspunkt gefunden wurde. FL$ = "T" bedeutet, er wurde gefunden, FL$ = "F" bedeutet, er wurde nicht gefunden. Die Variable PR speichert das vorherige Pixel. Wenn dieses Pixel Vordergrundfarbe hatte, ist PR = 0. Hatte das Pixel Hintergrundfarbe, ist PR = 1. Das folgende Beispiel verdeutlicht die Relevanz dessen.



In Spalte 1 wollen wir die Pixels zwischen Reihe 1 und Reihe 4 einzeichnen. Obwohl der obere Teil des Umrisses in dieser Spalte 2 Pixels hoch ist, soll er trotzdem als Anfangspunkt betrachtet werden. Dies geschieht durch das Verfolgen des vorherigen Pixels. Wenn das aktuelle Pixel 0 (Hintergrundfarbe) entspricht, das vorherige Pixel aber 1 (Vordergrundfarbe) entsprach, wissen wir, daß wir gerade über den Umriß hinausgegangen sind. Wir warten, bis wir den Umriß passiert haben, bevor wir mit dieser Information weiterarbeiten. Wenn der Umriß überschritten ist, gibt es zwei Möglichkeiten. Entweder ist der Umriß ein Anfangspunkt oder es ist ein Endpunkt. Ist der Umriß ein Anfangspunkt, erhält Y1 den aktuellen Wert von Y. Dies speichert den Anfangspunkt, bis er benutzt werden kann. Für die Variable FL$ würde "T" eingesetzt, da der Anfangspunkt gefunden wurde. Wenn der Umriß ein Endpunkt ist, zeichnet GOSUB 99 die

Pixels zwischen Y1 und der aktuellen Y-Koordinate. Für FL$ wird "F" eingesetzt, um nach einem weiteren Anfangspunkt zu suchen.

Es gibt noch einen speziellen Fall, den wir auch beachten müssen. Wenn wir einen Endpunkt gefunden haben, das untere Ende der Form aber erreicht haben, bevor dieser Punkt passiert wurde, müssen wir ihn trotzdem behandeln, als ob wir ihn passiert hätten.

Zeile 99 erledigt das eigentliche Ausmalen. GOSUB 70 schickt den Computer zur Unterroutine PLOT A POINT. ON ABS(RND(1)<PC) GOSUB 70 zeichnet einen Punkt in der angegebenen prozentualen Häufigkeit.

Weitere Informationen über Farben

Bis zu diesem Punkt haben wir das Entwerfen von Bildern mit Blöcken in Vorder-/Hintergrundfarben behandelt. Sie haben gelernt, daß bestimmte Farbkombinationen besser zusammenpassen als andere, daß Komplementärfarben einen scharfen Kontrast erzeugen, während andere Kombinationen verschwommene Bilder erzeugen können. Bestimmte Farbkombinationen können selbst emotionale Reaktionen hervorrufen, warme Farben (Rot, Orange, Gelb) ein Gefühl der Spannung, kalte Farben (Blau, Grün, Purpur) ein Gefühl der Ruhe. In diesem Abschnitt gehen wir noch näher darauf ein, wie Farben für visuelle Effekte eingesetzt werden können. Das Thema lautet: Schatten und sogenannte "Lichthöhen".

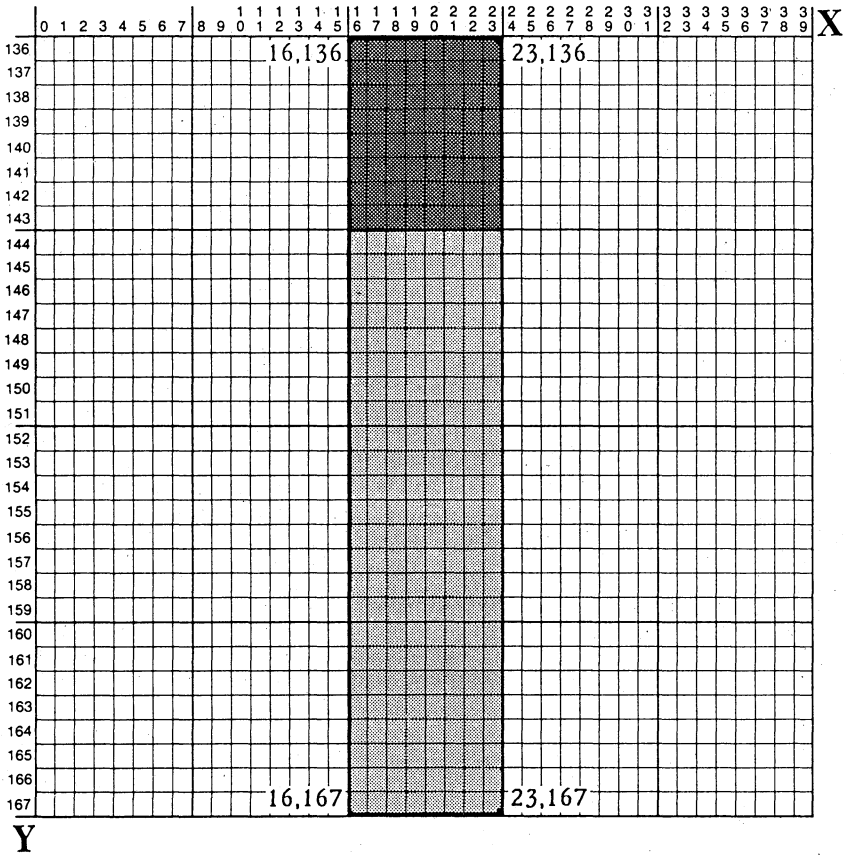
In der Malerei werden Schatten normalerweise durch Abdunkeln einer Farbe erzeugt. Die ursprüngliche Farbe wird Grundfarbe genannt. Diese Grundfarbe wird abgedunkelt, indem eine dunklere Farbe hinzugemischt wird - wie Purpur oder Schwarz. Beim **Commodore 64** müssen Sie sich nicht mit dem Mischen von Farben befassen, sie sind bereits vorhanden. Ein Schatten kann erzeugt werden, indem Sie zwei ähnliche Farben benutzen, wobei eine Farbe etwas dunkler als die Grundfarbe ist. Die dunklere Farbe sollte aber mit der Grundfarbe verwandt sein - beispielsweise sollten beide zu einer Farbgruppe gehören (warm oder kalt).

Sie können auch Lichthöhen in Ihre Bilder einarbeiten. Die Farbe für solch eine Lichthöhe ist fast dieselbe wie die Grundfarbe, nur etwas heller. Stellen Sie sich eine Lichtreflektion auf einem roten Ball vor. Der Bereich, in dem das Licht vom Ball reflektiert wird, wäre die Lichthöhe und würde hellrot erscheinen. Die Lichthöhe auf einem blauen Ball wäre hellblau, dieselbe auf Grün: hellgrün. Ein anderer Name für Lichthöhen lautet Weißaufhellung, d.h. Weiß wird einer Grundfarbe beigemischt.

Die Verwendung von Schatten und Lichthöhen in einem Bild vermittelt den Eindruck der Tiefe. Dies wird einfach erreicht, indem eine Seite eines Gegenstandes mit hellerer oder dunklerer Farbe gemalt wird. Probieren wir die Schattenwirkung beim Leuchtturm in Ihrem Bild aus. Wir nehmen an, daß die linke Seite des Leuchtturms von der Sonne abgewandt ist, so daß sie etwas dunkler als die rechte Seite sein muß. Um die linke Seite zu schattieren, zeichnen wir drei dunkle Linien von der Spitze bis zum unteren Ende des Leuchtturms.

Schattieren des Leuchtturms

Das folgende Diagramm zeigt, an welcher Stelle sich der Leuchtturm auf Ihrem Bildschirm befindet:



Sie zeichnen drei Linien an der linken Seite des Leuchtturms hinunter. Die Koordinaten dieser Linien lauten:

16,136 bis 16,167

17,136 bis 17,167

18,136 bis 18,167

Unten sehen Sie die Programmzeilen, die den Leuchtturm schattieren. Zeile 3010 ändert den Farbcode zu hellgrau auf weiß. Die Zeilen 3020-3040 zeichnen die drei Linien. Es spielt keine Rolle, in welcher Reihenfolge Sie X1, Y1, X2 oder Y2 angeben. Solange sie vor GOSUB 80 angegeben werden, kann eine Linie eingezeichnet werden. Da jede Linie von derselben Anfangsreihe (Y1 = 136) zu derselben Endreihe (Y2 = 167) eingezeichnet wird, müssen Y1 und Y2 nur einmal angegeben werden. Zum Schluß plaziert Zeile 3050 die Farben grau auf schwarz in den obersten Block des Leuchturms. Geben Sie nun diese neuen Programmzeilen ein.

```
3000 REM:::::::::SHADE LIGHTHOUSE
3010 C = 241
3020 X1 = 16: X2 = 16: Y1 = 136: Y2 = 167:
     GOSUB 80
3030 X1 = 17: X2 = 17: GOSUB 80
3040 X1 = 18: X2 = 18: GOSUB 80
3050 POKE 18090,11
```

Geben Sie ein: GOSUB 20: RUN 3000 RETURN. Schauen Sie sich an, wie der Leuchtturm schattiert wird, als ob die Sonne rechts außerhalb des Bildschirmes stände. Nun erscheint das Bild wirklich realistisch. Wenn der Vorgang des Schattierens beendet ist, betätigen Sie die LEERTASTE. Sie benötigen vielleicht noch eine genauere Erklärung zu diesem Vorgang:

Programmzeile #	Anfang der Linie X1, Y1	Ende der Linie X2, Y2
3020	16,136	16,167
3040	17,136	17,167
3040	18,136	18,167

Sie können leicht erkennen, daß in der obigen Tabelle weder Y1 noch Y2 je geändert wird. Deshalb müssen sie nur einmal im Programm angegeben werden. X1 und X2 ändert sich in jeder Zeile, also müssen diese neuen Spaltenangaben vor jedem GOSUB 80 eingegeben werden.

Schattierte und aufgehellte Gegenstände in einem Bild erzeugen einen spannungsreicheren Effekt, da sie den Eindruck der Tiefe und Plastizität vermitteln. Probieren Sie die Wirkung der Schatten und Lichthöhen in einer Skizze auf Millimeterpapier aus, bevor Sie das Programm eingeben. So können Sie schnell Änderungen vornehmen und lernen, die Farben für ein wirkungsvolles Bild zu bestimmen.

Beenden des Programms

Mit der Ausicht auf zwei weitere Kapitel, mag "Beenden des Programms" etwas voreilig klingen. Andererseits. dürften Sie auch bemerkt haben, daß das bisher erstellte Programm langsam etwas unhandlich wird. Um das Programm-Listing auf eine überschaubare Länge zu begrenzen, wird das Bild, das am Ende dieses Buches entstehen soll, mit drei separaten Programmen gezeichnet. Das erste dieser Programme beenden Sie an dieser Stelle.

Wie soll dieses "Bild mit drei Programmen" nun entstehen? Ihr Bild ist im Moment auf dem hochauflösenden Bildschirm gespeichert. Dort bleibt es, bis Sie: a) den Computer ausschalten oder b) ein anderes Programm starten, das alles auf dem Bildschirm löscht oder etwas daran ändert. Solange das Programm aus Kapitel 5 nichts bereits Gezeichnetes löscht oder ändert, wird das, was das Programm zeichnet, in Ihr Bild eingefügt. Also können wir das Bild aus Kapitel 4 auf den hochauflösenden Bildschirm holen, mit dem Programm aus Kapitel 5 weitere Details in das Bild einfügen und es mit dem Programm aus Kapitel 6 beschließen.

Am Anfang des Kapitels 5 laden Sie zuerst das Bild dieses Kapitels auf den hochauflösenden Bildschirm. Dann geben Sie die neuen Zeilen der Hauptroutine aus Kapitel 5 ein. Zum Schluß starten Sie das Programm aus Kapitel 5, um ein neues Objekt zu Ihrem Bild hinzuzufügen.

Wenn Sie Bilder kombinieren, die von zwei oder mehr Programmen gezeichnet werden, dominiert das zuletzt gestartete Programm. Das bedeutet, daß es sich überschneidende Bereiche kontrolliert. Angenommen Sie starten beispielsweise ein Programm, das ein weißes Haus auf hellblauem Hintergrund zeichnet. Dann starten Sie ein Programm, das eine rote Sonne auf gelbem Hintergrund zeichnet. Das

kombinierte Bild würde ein weißes Haus, eine rote Sonne und einen gelben Hintergrund zeigen (den Hintergrund, den das zuletzt gestartete Programm bestimmt).

Um dieses Programm zu beenden und fortzufahren, müssen noch einige Feinheiten an Ihrem Bild geändert werden. Eine Umwandlung betrifft den Rand des Bildes. Haben Sie gemerkt, daß das Wasser auf der einen Seite aus dem Nichts erscheint und auf der anderen Seite ebenso im Nichts zu verschwinden scheint? Dies liegt daran, daß in der Umrandung, die den hochauflösenden Bildschirm umgibt, nichts eingezeichnet werden kann. Um das Aussehen Ihres Bildes zu verbessern, zeichnen wir drei Linien als Begrenzung ein. Geben Sie Programmzeilen ein, die:

1. eine Linie an der linken Seite des Bildschirms hinunter einzeichnen, die genau bis zum Landstück reicht (Y = 159);
2. eine Linie ganz oben auf dem Bildschirm über die gesamte Breite einzeichnen;
3. eine Linie an der rechten Seite des Bildschirms hinunter einzeichnen, die genau bis zum Landstück reicht (Y = 166).

Zeichnen Sie diese Linie mit grünen Vordergrundpixels auf einem blauen Hintergrund ein. Beginnen Sie mit Zeile 4100

REM:::::::::DRAW BORDER.

Die Lösung sehen Sie unten. Sie können keine Umrandungslinien um das Wasser zeichnen, da dies 3 Farben (grün, schwarz und blau) in einigen der Blöcke des Wassers erfordern würde. Da Sie wissen, daß nur 2 Farben in einem Block verwendet werden können, zeichnen Sie die Umrandung nur bis zum Wasser hinunter.

```
4100 REM:::::::::DRAW BORDER
4110 C = 94: REM COLOR = GREEN ON BLUE
4120 X1 = 0: Y1 = 159
4130 X2 = 0: Y2 = 0: GOSUB 80
4140 X1 = 319: Y1 = 0: GOSUB 80
4150 X2 = 319: Y2 = 166: GOSUB 80
```

Sichern eines Bildes auf Diskette oder Band

In den Kapiteln 5 und 6 geben wir zwei neue und unabhängige Programme ein. Indem wir die Bilder aus diesem Kapitel und den Kapiteln 5 und 6 kombinieren, vollenden wir das Kunstwerk, das in diesem Buch behandelt wird. Im Moment gibt es nur eine Möglichkeit, Bilder aus zwei oder mehr Programmen in der Form zu

kombinieren, daß wir jedes Programm laden und starten. Sie werden aber feststellen, daß dies recht zeitaufwendig ist.

An dieser Stelle ist ein Werkzeug eingebaut, das ein Bild sichert. Dieses Werkzeug sichert das Bild, welches von einem Programm erstellt wurde. Das Bild kann jederzeit leicht und schnell in den Speicher geladen werden. Bevor Sie das Programm dieses Kapitels abspeichern, geben Sie sorgfältig die folgenden neuen Zeilen ein:

```
100 REM:::::::::SAVE PICTURE
101 INPUT "ENTER FILENAME"; FILE$
102 INPUT "ENTER 8 FOR DISK, OR 1 FOR
    CASSETTE"; DE
103 SYS 57812 FILE$ + ".PIC", DE
104 POKE 174,64: POKE 175,127: POKE 193,0:
    POKE 194,96
105 SYS 62954
106 SYS 57812 FILE$ + ".COL", DE
107 POKE 174,232: POKE 175,71: POKE 193,0:
    POKE 194,68
108 SYS 62954: END
```

Dieses Werkzeug ist eine "Routine" und wird in derselben Weise ausgeführt, wie die ZAP-Routine. Deshalb nehmen Sie sich die Zeit und sichern jetzt das Programm aus Kapitel 4. Starten Sie nie diese Routine oder die ZAP-Routine, bevor die aktuelle Version Ihres Programms sicher auf Diskette oder Band abgespeichert ist.

Sehen Sie sich die neue Routine noch einmal an. Wenn Sie dieses Werkzeug testen, können Sie es nur verbessern und noch einmal testen, indem Sie das Programm aus Kapitel 4 noch einmal laufen lassen. Das würde etwa 20 Minuten Zeit kosten. Nehmen Sie sich lieber jetzt 5 Minuten Zeit, um das Programm auf seine Richtigkeit hin zu überprüfen.

Um dieses Werkzeug zu benutzen, vergewissern Sie sich, daß sich eine Diskette in Ihrem Laufwerk befindet. Wenn Sie einen Cassettenrecorder benutzen, gehen Sie so vor, als ob Sie ein Programm sichern würden. Schreiben Sie den Dateinamen "KAPITEL4.PIC" auf Ihre Liste der Programmbezeichnungen. Notieren Sie die Anfangszahl vom Zählwerk. Sichern eines Bildes auf Band unterscheidet sich nicht vom Sichern eines Programms auf Band. Das Bild belegt einen bestimmten Platz auf dem Band und sollte nicht überspielt werden.

Um das Bild zu sichern, das sich im Moment im Speicher befindet, geben Sie ein: RUN 100 RETURN. Der Computer antwortet mit der Meldung "ENTER FILENAME", worauf Sie den Dateinamen

eingeben müssen, den das Bild erhalten soll. Der Name darf höchstens 12 Zeichen lang sein. Es ist wichtig, daß Sie verstehen, daß der Computer Ihr Bild in zwei verschiedenen Dateien speichert. Die erste Datei enthält die Pixelmuster, aus denen das Bild besteht, die zweite die Farben des Bildes. Jede hat einen Dateinamen, der mit den Zeichen beginnt, welche auch immer Sie nun eingeben. Für die eine Datei wird jedoch immer ".PIC" an den Namen angehängt, für die andere Datei ".COL". Geben Sie als Dateinamen für dieses Bild "KAPITEL4" ein.

Der Computer antwortet nun mit "ENTER 8 FOR DISK, OR 1 FOR CASSETTE". Wenn Sie mit einem Diskettenlaufwerk arbeiten, geben Sie 8 ein, andernfalls 1. Da die Meldungen von nun an dieselben sind, wie beim Sichern eines Programms, können Sie diese wie gewohnt beantworten.

Es gibt keine Möglichkeit, direkt zu kontrollieren, ob das Bild auf Ihrer Diskette/Ihrem Band gespeichert wurde. Um dies herauszufinden, müssen Sie den hochauflösenden Bildschirm löschen und dann das Bild laden. Bewegen Sie den Cursor auf eine leere Zeile und geben Sie ein:

```
C = 14: GOSUB 40: GOSUB 50 RETURN
```

So wird jede Abbildung auf dem Bildschirm gelöscht und alle Blöcke erhalten die Farben schwarz auf blau. Wenn der Cursor wieder erscheint, versuchen Sie das Bild zu laden. Dazu müssen Sie die folgenden zwei Dateien laden (eine direkt nach der anderen):

Für Benutzer von Diskettenlaufwerken

```
LOAD "KAPITEL4.PIC",8,1
```

dann

```
LOAD "KAPITEL4.COL",8,1
```

Für Benutzer von Cassettenrecordern

```
LOAD "KAPITEL4.PIC",1,1
```

dann

```
LOAD "KAPITEL4.COL",1,1
```

(Wenn Sie einen Cassettenrecorder benutzen, vergewissern Sie sich, daß das Band bis vor den Anfang des Bildes zurückgespult ist.)

Die obigen Dateien werden fast genauso geladen, wie Programmdateien, Sie müssen am Ende nur noch ",1" beifügen. Mit der 1 wird das Bild in Bank 1 an Stelle von Bank 0 aufgerufen, unabhängig davon, ob Sie ein Diskettenlaufwerk oder einen Cassettenrecorder benutzen.

Nachdem Sie den Befehl für eine dieser Dateien eingegeben haben, antwortet der Computer so, als ob Sie eine Programmdatei laden würden. Beantworten Sie deshalb alle Meldungen so, als ob Sie irgendeine andere Datei laden würden. (Es kann passieren, daß der Computer nicht mehr arbeitet, nachdem Sie die erste Bilddatei geladen haben. Betätigen Sie dann RUN/STOP und gleichzeitig RESTORE. Sie erhalten wieder Textanzeige und können die zweite Bilddatei laden.) Wenn beide Dateien geladen sind, geben Sie GOSUB 20 ein und betätigen Sie RETURN, um das Bild zu sehen.

Wenn das Bild aus Kapitel 4 nicht auf dem Bildschirm erscheint, betätigen Sie RUN/STOP RESTORE. Laden Sie das Programm aus Kapitel 4. Überprüfen Sie jede Zeile in der Routine SAVE PICTURE. Sehen Sie sich die Zeichen 0 und 1 an. Vergewissern Sie sich, daß Sie nicht versehentlich O oder l benutzt haben. Sehen Sie sich die Zeilennummern an. Haben Sie Zeilen ausgelassen? Prüfen Sie, ob Sie Semikolons (;) und Doppelpunkte (:) richtig eingesetzt haben. Wenn die Routine korrekt eingegeben ist, sichern Sie das Programm dieses Kapitel noch einmal (Informationen über das erneute Abspeichern eines Programms finden Sie in Kapitel 1) und starten Sie das Programm. Wenn das Bild wieder auf dem hochauflösenden Bildschirm erscheint, starten Sie die Routine SAVE PICTURE noch einmal.

> Beachten Sie: Das Laden von Bilddateien hat keine Wirkung auf ein Programm-Listing, das sich im Speicher befinden könnte, sehr wohl hingegen auf den hochauflösenden Bildschirm. Das neu geladene Bild löscht alles, was sich auf dem hochauflösenden Bildschirm befindet.

Werkzeug 100 :::::::::: Abspeichern eines Bildes

```
100 REM::::::::::SAVE PICTURE
101 INPUT "ENTER FILENAME"; FILE$
102 INPUT "ENTER 8 FOR DISK, OR 1 FOR
    CASSETTE"; DE
103 SYS 57812 FILE$ + ".PIC", DE
```

```
104 POKE 174,64: POKE 175,127: POKE 193,0:
    POKE 194,96
105 SYS 62954
106 SYS 57812 FILE$ + ".COL", DE
107 POKE 174,232: POKE 175,71: POKE 193,0:
    POKE 194,68
108 SYS 62954: END
```

Zweck: *Diese Routine sichert Ihr Bild auf Diskette oder Band, so daß es schnell auf den hochauflösenden Bildschirm gerufen werden kann. Das Bild wird in zwei Dateien abgespeichert, von denen die eine das Bildmuster, die andere das Farbmuster enthält.*

Anwendung: *Geben Sie RUN 100 ein und betätigen Sie RETURN, um das Bild zu sichern, das sich gerade im Computerspeicher befindet. Sie werden nach einem Dateinamen gefragt. Sie können jeden gewünschten Namen (bis zu einer Länge von 12 Zeichen) eingeben. Nach der Eingabe des Dateinamens werden Sie nach der Nummer des Speichergerätes gefragt. Geben Sie 8 ein, wenn Sie ein Diskettenlaufwerk benutzen, oder 1, wenn Sie einen Cassettenrecorder benutzen. Das Bild wird dann in derselben Weise abgespeichert, wie ein Programm. Um ein Bild wieder in den Computerspeicher zu laden, geben Sie ein:*

LOAD "Dateiname.PIC",8,1 (Diskette) LOAD "Dateiname.COL",8,1 oder
LOAD "Dateiname.PIC",1,1 (Band) LOAD "Dateiname.COL",1,1

Beachten Sie, daß "Dateiname" durch den Dateinamen ersetzt werden muß, den Sie Ihrem Bild gegeben haben, als Sie es gesichert haben.

Technische Beschreibung: *Diese Routine benutzt drei Bestandteile des BASIC-Befehls "SAVE", um das Bild zu sichern. Diese Bestandteile werden benutzt, um:*

1. *den Namen in Anführungszeichen und die Nummer des Speichergerätes anzugeben;*
2. *den Speicherbereich, der die zu sichernden Informationen enthalten soll, zu bestimmen;*
3. *die Informationen im angegebenen Bereich abzuspeichern.*

Die Zeilen 101 bis 103 sammeln den Dateinamen und die Nummer des Speichergerätes. SYS 57812 benutzt einen Teil des Befehls SAVE, um den eingegebenen Dateinamen und die Gerätenummer zu sichern. Zeile 104 bestimmt den Speicherbereich, der das Pixelmuster enthalten soll, und Zeile 105 sichert die Information in dem Bereich. Die Zeilen 106 bis 108 wiederholen diesen Vorgang, um die Farbinformationen auf Diskette/Band abzuspeichern.

Zusammenfassung

Sie haben den ersten wichtigen Meilenstein in diesem Buch erreicht. Da das erste Programm vollständig ist, kosten die letzten beiden Kapitel kaum noch Zeit. Selbst wenn Sie nun das Buch nicht mehr weiterlesen sollten, ist Ihr Wissen über Computergrafik groß genug, um:

- Linien in jede Richtung zu zeichnen,
- Linien zu Umrissen von Formen zu verbinden,
- Formen jeder Größe zu zeichnen,
- Schatten und Lichthöhen zu verwenden, um Ihren Bildern ein realistischeres Aussehen zu geben,
- Bilder auf Diskette/Band zu sichern und sie wieder abzurufen.

Das Zeichnen von Linien und Formen ist einfach und macht Spaß. Eine Linie ist einfach eine Reihe von hintereinander eingezeichneten Pixels. Wenn Sie die Routine PLOT A LINE einsetzen, müssen Sie nur den Anfangs- und Endpunkt jeder Linie kennen.

Linien können in jede Richtung gezeichnet werden und an jeder Stelle auf dem Bildschirm beginnen. Das Einzeichnen eines Punktes außerhalb des Bildschirms kann bewirken, daß der Computer nicht mehr arbeitet. Denken Sie daran, daß X-Koordinaten die Spalte angeben und von 0 bis 319 reichen. Y-Koordinaten geben die Reihe an und reichen von 0 bis 199.

Durch Verbindung mehrerer Linien können Sie die Umrisse von Formen bilden. Dies ist die erste Stufe, Formen farbig zu gestalten. Umrisse werden von oben nach unten (H = ?), und von links nach rechts (W = ?) ausgemalt. Da jede Spalte ganz durchlaufen wird, schalten fortlaufende Bestimmungen von Umrißpixels das Einzeichnen ein und aus. Dies müssen Sie sich merken, wenn Sie Formen entwerfen, die später ausgemalt werden sollen.

Farben können verschiedene visuelle Effekte erzeugen - wie Schatten (dunkler als die Grundfarbe) und Lichthöhen (heller als die Grundfarbe). In einer Skizze auf Millimeterpapier können Sie schnell erkennen, welche Farbkombinationen gut zusammenpassen und welche

Farbblöcke betroffen sind. Wenn Sie sich für die Farben entschieden haben, können Sie für C den entsprechenden Farbcode für den Vorder/Hintergrund einsetzen.

In Kapitel 5 lernen Sie, wie Sie ein neues Werkzeug mit dem Namen DRAW A SHAPE benutzen können. Dieses Werkzeug ermöglicht es Ihnen, eine Form einmal zu zeichnen und sie dann an irgendeiner Stelle auf dem Bildschirm und so oft Sie wollen zu vervielfältigen. Wenn Sie vorhaben, detaillierte Computerbilder zu erstellen, werden Sie sehen, daß dieses Werkzeug als Bestandteil Ihres Werkzeugkastens von unschätzbarem Wert ist.

Kapitel 5

Probezeichnen und Vervielfältigen von Formen

Bis jetzt haben Sie Punkte und Linien eingezeichnet und Formen ausgemalt. Nun sind Sie auch in der Lage, das Schiff in Ihr Bild einzuzeichnen. Dazu müssen Sie 68 Linien zeichnen. Darüberhinaus bereichern Sie Ihr Bild noch mit einigen Seemöven. Dies führen Sie alles mit Hilfe eines neuen Werkzeugs unter der Bezeichnung DRAW A SHAPE aus.

Dieses Werkzeug übernimmt zwei nützliche Aufgaben. Erstens bietet es eine einfache Möglichkeit, eine Form probeweise auf dem Bildschirm einzuzeichnen, sich anzuschauen, wo sie in Ihrem Bild erscheint, und das Programm zu verändern, wenn die Form noch in eine Richtung verschoben werden muß. Mit dem Werkzeug PLOT A LINE aus dem letzten Kapitel hatten Sie nur die Möglichkeit, eine falsch eingezeichnete Form zu verschieben, indem Sie jede X- und Y-Koordinate für jede Linie veränderten. Bei diesem neuen Werkzeug muß nur einmal X und Y verändert werden, um die ganze Form an einer anderen Stelle neu einzuzeichnen. Das Schiff wurde mehrmals probeweise eingezeichnet, bevor seine endgültige Position feststand. Sie werden sehen, daß die Möglichkeit des Probezeichnens eine Menge Zeit bei der Erstellung von Formen jeder Art spart.

Das zweite wichtige Charakteristikum dieses Werkzeugs ist sein Anwendungsbereich in der Vervielfältigung von Formen. Wenn eine Form mit diesem Werkzeug einmal eingezeichnet ist, können Sie sie so oft Sie wollen an beliebiger Stelle auf dem Bildschirm vervielfältigen. Dazu müssen Sie nur angeben, wo die Kopie(n) erscheinen soll(en). Stellen Sie sich vor, wieviel Zeit Sie beim Programmieren sparen können, wenn:

- aus einer Person eine Menschenmenge,
- aus einem Auto ein Verkehrschaos,
- aus einem Haus eine Stadtszene,
- aus einem Ziegelstein eine ganze Mauer,
- aus einer Blume ein herrlicher Blumenstrauß entsteht

All dies wird durch das Aufbewahren einer schriftlichen Beschreibung Ihrer Form im Computerspeicher erreicht. Jedes Mal, wenn die Form auf dem Bildschirm verschoben oder vervielfältigt werden muß, sieht der Computer sich die gespeicherte Beschreibung an.

Definition einer Form mittels Datenlisten

Die Beschreibung der Form besteht aus im Speicher befindlichen Listen, die zum einen die "Endpunkte" der Form, zum anderen die Linien der Form beschreiben. Als erstes müssen Sie die Informationen, die die zu zeichnende Form beschreiben, sammeln. Diese beiden Listen können leicht zusammengestellt werden, wenn Sie die Form auf Millimeterpapier skizzieren.

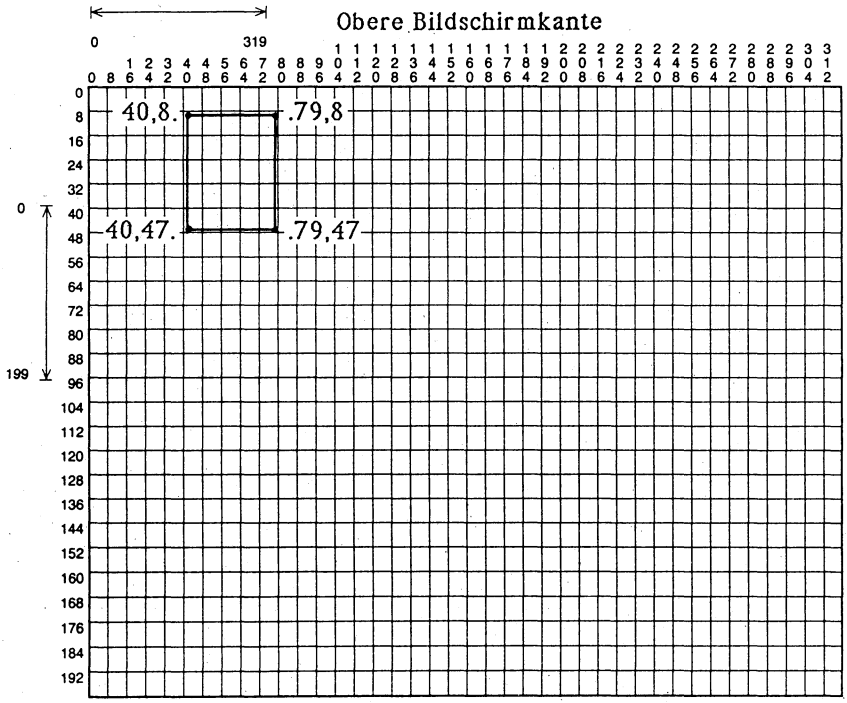
In der ersten Liste notieren Sie alle X,Y-Koordinaten, die Endpunkte in der Form bilden. Jeder Punkt, der sich am Anfang oder am Ende einer Linie befindet, ist ein "Endpunkt". Ein Quadrat hat also vier Endpunkte, ein Dreieck hat drei Endpunkte. Die Liste der Endpunkte sollte folgendermaßen aussehen:

	Endpkt.0	Endpkt.1	Endpkt.2	Endpkt.3	...Endpkt.N
X					
Y					

Jedem Endpunkt ist eine Nummer zugewiesen, die sie über jeder Spalte sehen. Es spielt keine Rolle, welchem Endpunkt Sie welche Nummer zuweisen, denn solange Sie bei der Nummer 0 beginnen und keine Nummer überschlagen (d.h. 0,1,2,etc.), werden keine Probleme auftreten. Wir nehmen als Beispiel ein Quadrat, wohlwissend daß der Begriff "Quadrat" hier nicht ganz zutreffend ist. Eigentlich ist der Abstand zwischen jeder Reihe auf dem Bildschirm etwa 1,234 mal größer als der Abstand zwischen jeder Spalte. Für ein richtiges Quadrat müßten also 1,234 mal mehr Spalten in der Breite als Reihen in der Höhe des Quadrats eingezeichnet werden.

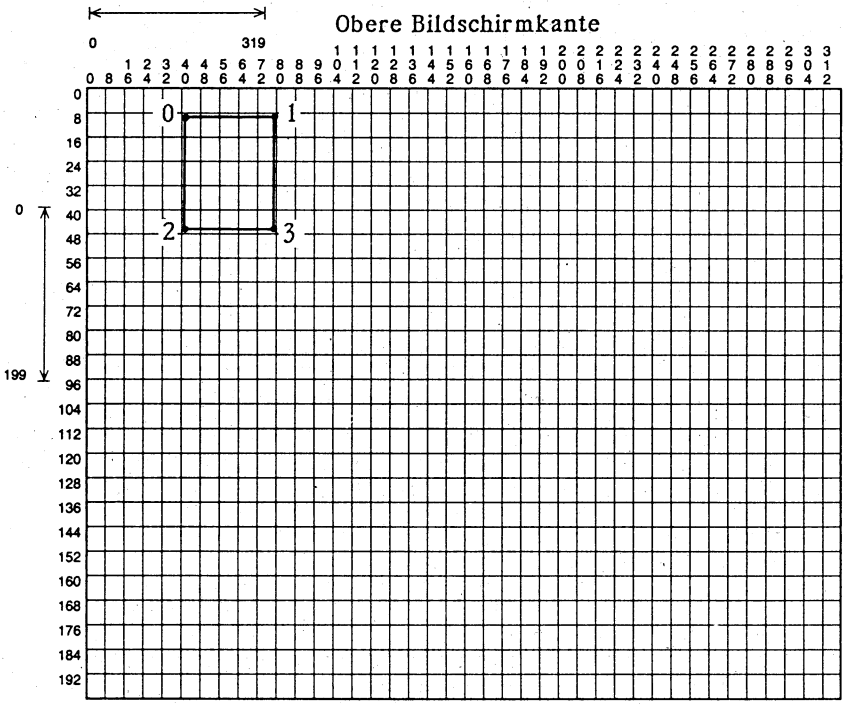
Wir schlagen nachfolgendes Vorgehen für die Herstellung einer Endpunkte-Liste vor:

1. Skizzieren Sie die Form auf Millimeterpapier und notieren Sie die X,Y-Koordinaten für jeden Endpunkt:



X, Y Pixelpunkte

2. Geben Sie jedem Punkt eine Erkennungsnummer(#), bei 0 beginnend:



X, Y Pixelpunkte

3. Tragen Sie die X,Y-Koordinaten für jeden Endpunkt unter der richtigen Spaltennummer in Ihre Liste ein:

	Endpkt.0	Endpkt.1	Endpkt.2	Endpkt.3
X	40	79	40	79
Y	8	8	47	47

Dieser Schritt ist sehr wichtig. Vergewissern Sie sich, daß Sie die X,Y-Koordinaten für den Endpunkt #0 in der ersten Spalte, die für den Endpunkt #1 in der zweiten Spalte, die für den Endpunkt #2 in der dritten Spalte, usw. eintragen. Die X-Koordinate steht immer in der ersten, die Y-Koordinate in der zweiten Reihe.

Nun haben Sie alle Informationen für die erste Liste, die wir im folgenden "Endpunktdaten"-Liste nennen wollen. In komplexen Formen mit vielen Endpunkten kann diese Liste sehr lang werden. Deshalb sollten Sie immer Überprüfen, ob jeder Endpunkt einer Form in Ihrer Liste richtig erscheint, bevor Sie fortfahren.

Bis jetzt haben Sie nur die Endpunkte der Form notiert. Sie benötigen also noch eine weitere Liste, die Linie für Linie eine Beschreibung der Form gibt und folgendermaßen aussehen sollte:

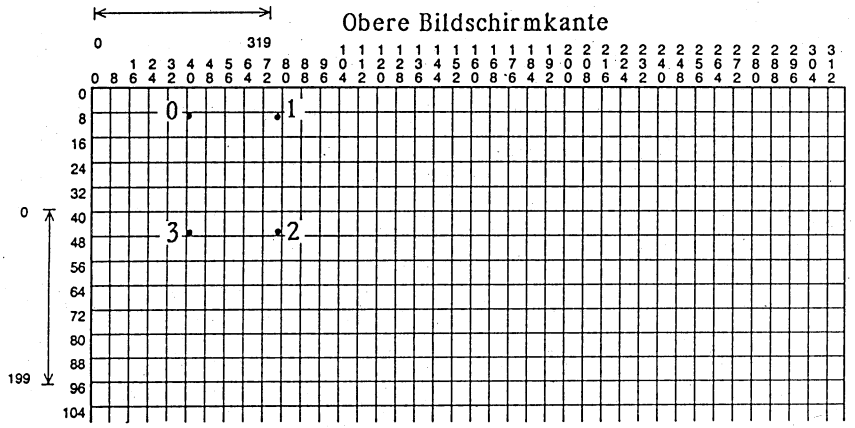
	Linie 0	Linie 1	Linie 2	Linie 3	...Linie N
"VON" Endpkt.#					
"ZU" Endpkt.#					

Sie benötigen für jede Linie der Form eine Spalte, um ihre "von/zu"-Endpunkte zu notieren. Alle Linien werden "von" einem Endpunkt "zu" einem weieren Endpunkt eingezeichnet. Beispielsweise benötigen Sie für eine Form mit 27 Linien 27 Spalten (numeriert von 0 bis 26). Die Nummern, die Sie jedem Endpunkt gegeben haben, tragen Sie in die Spalten für jede Linie der Form ein. Für das Quadrat bedeutet dies:

	Linie 0	Linie 1	Linie 2	Linie 3
"VON" Endpkt.#	9	1	2	3
"ZU" Endpkt.#	1	2	3	0

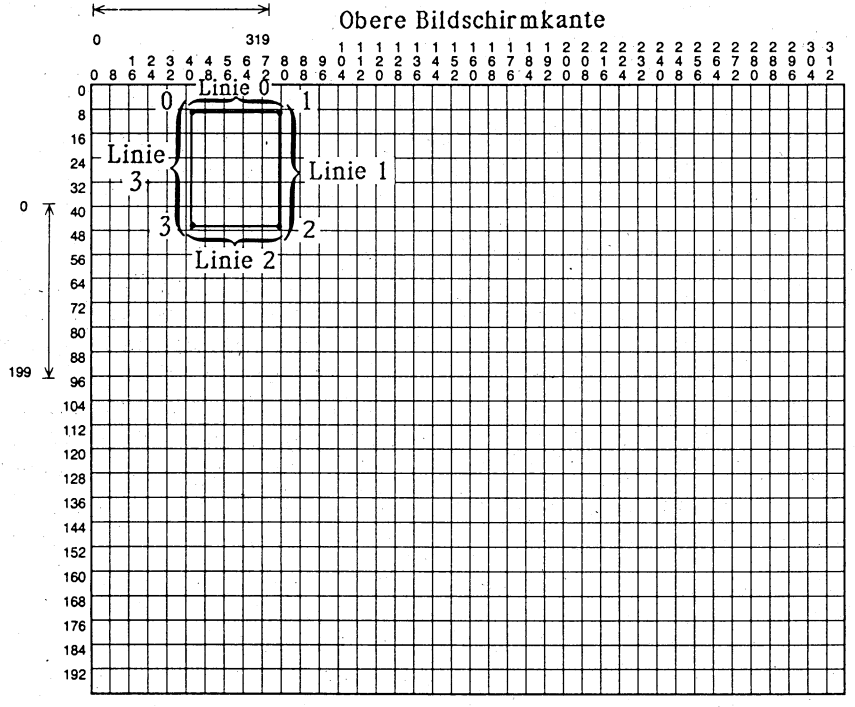
Auch hier spielt es keine Rolle, welche Linie in die Spalte #0, oder welche in die Spalte #1 eingetragen wird. Solange die Spalten bei 0 beginnend numeriert sind und keine Nummer überschlagen wird, treten keine Probleme auf. Wichtig ist aber, daß Sie die richtigen Nummern der Endpunkte aus der ersten Liste eintragen.

Die zweite Liste für die Unterroutine DRAW A SHAPE ist nun auch vollständig. Diese Liste nennen wir "Liniendaten". Sehen wir uns an, wie der Computer diese Listen benutzt, um eine Form nochmals oder neu zu zeichnen. Bei unserem Beispiel enthält die Liste der Endpunkt-Daten alle Endpunkte der Form:



X, Y Pixelpunkte

Mit Hilfe Ihrer Liste der Liniendaten können die Punkte leicht miteinander verbunden werden:

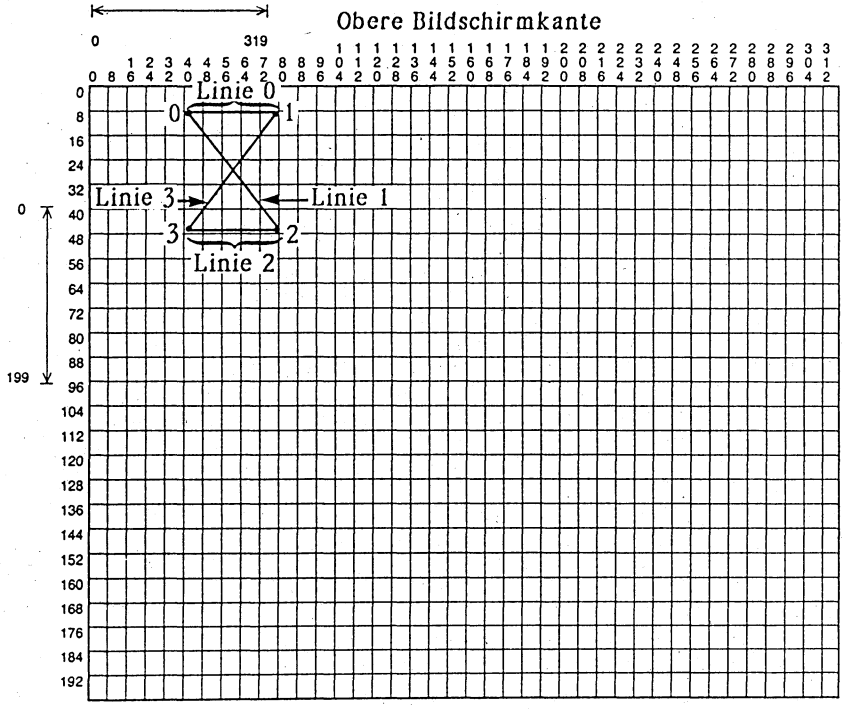


X, Y Pixelpunkte

Was würde passieren, wenn wir die Liste der Liniendaten folgendermaßen ändern würden:

	Linie 0	Linie 1	Linie 2	Linie 3
"VON" Endpkt.#	0	1	3	2
"ZU" Endpkt.#	1	3	2	0

Obwohl die Liste der Endpunktdaten dieselbe bleibt, beschreibt die Liste der Liniendaten folgende Form:



X, Y Pixelpunkte

Sobald Sie die beiden Datenlisten zusammengestellt haben, sind Sie schon ein gutes Stück weiter. Wenn Sie sie im Speicher Ihres **Commodore 64** ablegen, kann die Form eingezeichnet, bewegt und vervielfältigt werden, indem der Computer einfach auf die Listen sieht.

Als nächstes werden diese Listen in Ihr Programm eingegeben.

(Beachten Sie: Derartige Listen werden "Arrays" genannt. Wir aber bleiben, um Verwirrung zu vermeiden, weiterhin bei der Bezeichnung Listen.)

Eingabe von Datenlisten in das Programm

Die erste Liste, die in das Programm eingegeben wird, ist die der Endpunktdaten. Zuerst müssen Sie dem Computer die Größe der Liste angeben. Dazu haben Sie den Befehl DIM zur Verfügung, der die Anzahl der Reihen und Spalten der Liste angibt. Die Liste der Endpunktdaten für das Quadrat besteht aus 2 Reihen und 4 Spalten. Um dem Computer diese Abmessungen mitzuteilen, benötigen Sie folgende Hauptroutinenzeile:

```
1100 DIM E%(1,3)
```

Beachten Sie: Geben Sie während der Erörterung der Routine noch keine Programmzeilen ein. Dies erfolgt erst im nächsten Abschnitt "Probezeichnen".

Der Befehl DIM erfüllt mehrere Zwecke. Erstens gibt er der Liste einen Namen. Wir haben den Namen "E" für Liste der Endpunkte gewählt. Das Zeichen "%" teilt dem Computer mit, daß die Liste benutzt wird, um ganze Zahlen zu speichern. Denken Sie beim Speichern der Endpunktdaten-Liste immer daran, daß die Unterroutine DRAW A SHAPE die Angabe von E% erwartet.

Die Zahlen in Klammern werden Indizes genannt. Sie geben die Anzahl der Reihen und Spalten an, die die Liste belegt. Die erste Zahl steht für die Anzahl der Reihen in der Liste. Die Numerierung der Reihen beginnt bei 0, also bedeutet 1, daß die Liste 2 Reihen belegt. Die zweite Zahl steht für die Anzahl der Spalten in der Liste, so daß, ebenfalls von 0 ausgehend, die Ziffer 3 bedeutet, daß die Liste 4 Spalten belegt.

Anhand der Größenangabe kann der Computer genug Speicherplatz (nicht mehr und nicht weniger) für die Speicherung der Daten aus der Liste reservieren. Wenn Sie angeben, daß Ihre Liste nur 10 Spalten belegt und versuchen dann 11 Spalten einzutragen, erhalten Sie die Meldung "BAD SUBSCRIPT ERROR".

Keine Probleme macht Ihnen der Computer demgegenüber, wenn Sie die Liste länger angeben als notwendig. Es ist in der Mehrzahl der Fälle durchaus nützlich, wenn der Computer mehr Speicherplatz reserviert, als Ihre Liste tatsächlich benötigt. Die Liste E% wird nie mehr als zwei Reihen (eine für die X- und eine für die

Y-Koordinate) belegen, deshalb kann die Größenangabe 1 bleiben. Die Anzahl der benötigten Spalten hängt jedoch von der Anzahl der Endpunkte in Ihrer Form ab. Wir schlagen vor, daß Sie E% mit 100 Spalten erstellen, die für fast alle Formen ausreichen. Um für E% diese Größe anzugeben, muß die Programmzeile lauten:

```
1100 DIM E%(1,99)
```

Wenn diese Zeile in das Programm eingegeben ist, haben Sie eine leere E%-Liste, die mit Endpunktdaten ausgefüllt werden muß. Vergleichen wir einmal die leere Liste im Speicher mit der handgeschriebenen Liste der Endpunktdaten:

Leere E% - Liste im Speicher

	Spalte 0	Spalte 1	Spalte 2	Spalte 3	...Spalte 99
Reihe 0					
Reihe 1					

Liste der Endpunktdaten

	Endpkt. 0	Endpkt. 1	Endpkt. 2	Endpkt. 3
X	40	79	40	79
Y	8	8	47	47

Sie müssen die Möglichkeit haben, Ihre Liste direkt in den Speicher zu kopieren. Jede "Stelle" im Speicher kann mit einer Dateneinheit ausgefüllt werden, indem die richtige Reihen- und Spaltennummer verwendet wird. Beispielsweise ist die erste Stelle in E% Reihe 0, Spalte 0. Dies wird angegeben als E%(0,0). Die nächste Stelle darunter ist Reihe 1, Spalte 0. Dies wird als E%(1,0) angegeben. (Beachten Sie, daß die Reihenangabe immer vor der Spaltenangabe erfolgt.) Für die Daten des Quadrats wollen wir:

40 an die Stelle (0,0),

8 an die Stelle (1,0),

79 an die Stelle (0,1),

8 an die Stelle (1,1),

40 an die Stelle (0,2),

47 an die Stelle (1,2),

79 an die Stelle (0,3),

47 an die Stelle (1,3) setzen

Diese Zahlen werden folgendermaßen in die E%-Liste eingesetzt:

	Spalte 0	Spalte 1	Spalte 2	Spalte 3	...Spalte 99
Reihe 0	40	79	40	79	
Reihe 1	8	8	47	47	

Sehen Sie sich genau an, wie die Daten und Positionen in den Tabellen eingesetzt werden. Wenn Sie nicht verstehen, warum 40 an der Stelle (0,0) eingesetzt wurde und weiterhin, was es generell mit der Position (0/0) auf sich hat, werden Sie die Vorteile des Werkzeugs dieses Kapitels nicht richtig ausnutzen können. Schauen Sie sich die Tabellen wirklich solange an, bis Sie alles verstanden haben. Vielleicht sehen Sie sich auch noch einmal das ursprüngliche Quadrat mit seinen Koordinaten und Endpunktnummern an.

Um die Endpunktdaten an die richtigen Stellen in E% zu setzen, werden eine FOR/NEXT-Schleife und der Befehl DATA eingegeben:

```
1110 FOR I = 0 TO 3
1120 READ E%(0,I), E%(1,I)
1130 NEXT I
1140 DATA 40, 8, 79, 8, 40, 47, 79, 47
```

Zeile 1110 bildet den Anfang der Schleife und setzt I auf 0. Zeile 1120 liest (READ):

E%(0,0),E%(1,0)

Dies sind die beiden Stellen in Spalte 0 in E%. Die Daten des Endpunktes #0 müssen an diesen Stellen eingesetzt werden, also sind sie die ersten beiden Koordinaten, die in Zeile 1140 angegeben sind. Jedesmal wenn die Schleife durchlaufen wird, werden die nächsten beiden Dateneinheiten von Zeile 1140 in die Liste eingesetzt. Da I bei 0 beginnt und sich bis 3 fortsetzt, wird die Schleife viermal durchlaufen. Jedesmal ändert sich Zeile 1120, um eine weitere Spalte in der Liste anzugeben:E%(0,1), E%(1,1), E%(0,2), E%(1,2), etc. Für jeden Durchlauf wird ein weiteres Koordinatenpaar aus dem Befehl DATA gelesen und in eine Spalte eingesetzt.

Folgende wichtige Punkte müssen bei der FOR/NEXT-Schleife und dem Befehl DATA beachtet werden:

1. Für die Schleife (FOR I = 0 TO ?) muß die richtige Anzahl der Durchläufe bestimmt werden. Wenn mit dem Befehl DATA 20 Einheiten angegeben sind, muß für die Schleife FOR I = 0 TO 9 bestimmt werden. (Denken Sie daran, daß die Schleife zwei Dateneinheiten auf einmal liest.) Sie können unvollständige und unerwünschte Formen erhalten, wenn die Schleife nicht korrekt bestimmt wird.
2. Der Befehl READ füllt die Liste mit Daten, zwei Einheiten pro Durchgang (eine X- und eine Y-Koordinate). Er benutzt dabei die Variable "I", um jede Spaltenstelle in der Liste anzugeben. Solange Sie diesen Befehl READ nicht in einem Ihrer Programme ändern, füllt er immer die E%-Liste aus.
3. Um den Befehl DATA einzugeben, kopieren Sie die Liste der Endpunktdaten Ihrer Form. Kopieren Sie die Koordinaten Spalte für Spalte, beginnend bei Endpunkt #0. Die X-Koordinate muß immer vor der Y-Koordinate eingegeben werden. Für das Quadrat wird dies folgendermaßen ausgeführt:

```
                 Endpkt.0   Endpkt.1   Endpkt.2   Endpkt.3
                   X,Y,       X,Y,       X,Y,       X,Y
                    ^          ^          ^          ^
1140 DATA          40,8,      79,8,      40,47,     79,47
```

Da die Endpunkte von 0 bis 3 numeriert sind, wird auch die Schleife entsprechend bestimmt:

```
1110 FOR I = 0 TO 3
```

Soviel zur E%-Liste! Die Übertragung der Liste der Liniendaten ist genauso einfach. Sehen wir uns die Liste noch einmal an:

	Linie 0	Linie 1	Linie 2	Linie 3
"VON" Endpkt.#	0	1	2	3
"ZU" Endpkt.#	1	2	3	0

Die Größe und der Name dieser Liste müssen in das Programm eingegeben werden. Die neue Unterroutine erwartet, daß die Liste L% (für Liniendaten) genannt wird. Können Sie sich die Angaben für den Befehl DIM vorstellen? Die Liste benötigt 2 Reihen und wenigstens 4 Spalten, die kleinsten Ausmaße, die wir eingeben müßten, wären also:

```
1200 DIM L%(1,3)
```

Um genug Platz für andere Formen verschiedener Größen zu reservieren, werden für diese Liste ebenfalls 100 Spalten angegeben:

```
1200 DIM L%(1,99)
```

Mit folgenden Hauptroutinezeilen werden die Liniendaten in L% gespeichert:

```
1210 FOR I = 0 TO 3
1220 READ L%(0,I), L%(1,I)
1230 NEXT I
1240 DATA 0, 1, 1, 2, 2, 3, 3, 0
```

Für die Schleife wird ein Durchgang für jeden Satz von Dateneinheiten bestimmt - nicht mehr und nicht weniger. Wenn Ihre handgeschriebene Liste Linie #0 bis Linie #12 enthält, muß für die Schleife FOR I = 0 TO 12 bestimmt werden.

Zeile 1220 füllt L% mit den Dateneinheiten in Zeile 1240 aus. Bei jedem Durchlaufen der Schleife liest Zeile 1220 zwei Dateneinheiten aus dem Befehl DATA. (Beachten Sie: Wenn der Befehl DATA eine ungerade Zahl von Dateneinheiten enthält, stimmt etwas nicht.) Die Dateneinheiten werden Spalte für Spalte in die L%-Liste eingesetzt.

Zeile 1240 muß eine Kopie Ihrer Liste der Liniendaten sein und wird in folgender Weise zusammengestellt:

	Linie 0	Linie 1	Linie 2	Linie 3
	von,zu,	von,zu,	von,zu,	von,zu
1240 DATA	0,1,	1,2,	2,3,	3,0

Selbstverständlich sind alle Zeilennummern willkürlich gewählt. Sie können jede verfügbare Programmzeile in einem Programm, mit dem Sie gerade arbeiten, benutzen. Es kommt nur darauf an, wie Sie die Programmzeilen bestimmen.

Doch genug der Worte! Sie benötigen zum besseren Verständnis des Gesagten zunächst ein wenig Praxis.

Probezeichnen

Sie geben zuerst die Unterroutine DRAW A SHAPE ein und zeichnen dann eine Form probeweise ein. Da das neue Werkzeug mit

den Werkzeugen PLOT A POINT und PLOT A LINE arbeitet, laden Sie das Programm aus Kapitel 4 in den Speicher. Listen Sie die Zeilen 1000 bis 1400 auf dem Bildschirm.

Die ersten drei Zeilen der Hauptroutine sind notwendig, um die eingezeichnete Form zu sehen, während der Rest der Hauptroutine hier überflüssig ist. Eine kleine Änderung der ZAP-Routine erhalten die Zeilen 1100, 1110 und 1120, wenn Sie die Routine mit RUN 10 starten. Listen Sie die Zeilen 10 und 11.

Dies sind die ersten beiden Zeilen der ZAP-Routine. Zeile 11 lautet:

```
11 A = 256: B = 2049: C = 1003
```

"C = 1003" bestimmt, bei welcher Zeile der Löschvorgang beginnen soll. Geben Sie diese Zeile sorgfältig in folgender Form neu ein:

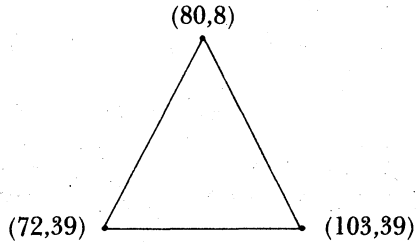
```
11 A = 256: B = 2049: C = 1200
```

Die kleine Änderung beläßt alle Zeilen vor Zeile 1200 im Programm. Geben Sie RUN 10 ein und RETURN.

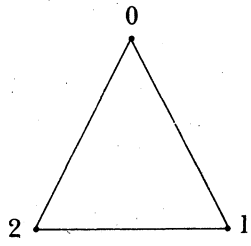
Warten Sie, bis die Hauptroutine gelöscht ist, dann listen Sie Ihr Programm, um die Unterroutinen zu überprüfen (es müssen mehrere sein). Kontrollieren Sie desgleichen, ob die Hauptroutine nur die Zeilen 1100, 1110 und 1120 enthält. Wenn etwas fehlt, laden Sie das Programm aus Kapitel 4 erneut und führen die oben angegebenen Schritte noch einmal aus. Ist alles in Ordnung, geben Sie die folgende Unterroutine DRAW A SHAPE ein:

```
110 REM:::::::::DRAW A SHAPE
111 FOR J = 0 TO NL
112 E1 = L%(0,J): E2 = L%(1,J)
113 X1 = E%(0,E1) + X0: Y1 = E%(1,E1) + Y0
114 X2 = E%(0,E2) + X0: Y2 = E%(1,E2) + Y0
115 GOSUB 80
116 NEXT J
117 RETURN
```

Auch hier sollten Sie, bevor Sie fortfahren, eine präzise Zeilenkontrolle vornehmen. Um sicherzugehen, daß Sie alle Zeilen korrekt eingegeben haben, müssen Sie eine Form zeichnen. Nehmen wir dieses Dreieck:



Um die beiden Datenlisten zusammenzustellen, geben Sie jedem Endpunkt eine Nummer:



Dann können Sie die Datenlisten einfach ausfüllen:

	Endpkt. 0	Endpkt. 1	Endpkt. 2
X	88	103	72
Y	8	39	39

(Koordinaten jedes Endpunktes in der Form)

	Linie 0	Linie 1	Linie 2
"VON" Endpkt.#	0	1	2
"ZU" Endpkt.#	1	2	0

(Linien "von" einem "zu" einem weiteren Endpunkt mit den oben aufgeführten Endpunktnummern)

Zur Eingabe dieser Listen in das Programm ist ein DIM-Befehl notwendig, der wie folgt auszusehen hat:

```
1200 DIM E%(1,99), L%(1,99)
```

Sie sehen, daß Sie die Ausmaße für beide Listen in derselben Programmzeile angeben können. Für jede bestimmen Sie 2 Reihen und 100 Spalten. Um E% mit den richtigen Endpunktkoordinaten auszufüllen, geben Sie ein:

```
1300 REM:::::::::TRIANGLE ENDPOINTS
1310 FOR I = 0 TO 2
1320 READ E%(0,I), E%(1,I)
1330 NEXT I
1340 DATA 88,   8, 103,  39,  72,  39
```

Die Endpunkte reichen von #0 bis #2, also wird für die Schleife 0 TO 2 bestimmt. Die Dateneinheiten aus der Endpunktliste sind in Zeile 1340 eingegeben, beginnend mit den X- und Y-Koordinaten des Endpunktes #0. Zeile 1320 liest die Dateneinheiten in die entsprechenden Stellen der Liste.

Geben Sie folgende Zeilen ein, um die Liste der Liniendaten zu erstellen:

```
1400 REM:::::::::TRIANGLE LINES
1410 FOR I = 0 TO 2
1420 READ L%(0,I), L%(1,I)
1430 NEXT I
1440 DATA 0, 1, 1, 2, 2, 0
```

Für die Schleife werden 3 Durchgänge bestimmt. (Beachten Sie: Die E%- und L%-Schleifen müssen für gewöhnlich unterschiedlich oft durchlaufen werden.) Zeile 1420 befiehlt dem Computer, Dateneinheiten in die L%-Liste zu lesen. Die Dateneinheiten erscheinen in Zeile 1440, beginnend mit den "von"/"zu" Endpunkten der Linie #0.

Folgende Zeilen geben Sie ein, damit die Unterroutine DRAW A SHAPE diese Listen benutzt:

```
1500 REM:::::::::DRAW TRIANGLE
1510 C = 14
1520 NL = 2: X0 = 0: Y0 = 0
1530 GOSUB 110
```

Zeile 1520 ist unbedingt notwendig, um eine Form probeweise zu zeichnen. Wir erklären dies noch, nachdem Sie das Programm getestet haben. Starten Sie es ersteinmal.

Sie sehen, daß der Bildschirm zu Schwarz auf Blau geändert wird und alle Pixels die Hintergrundfarbe erhalten. Nach einem Augenblick wird das Dreieck oben links eingezeichnet. Wenn es komplett ist

(oder wenn Probleme auftauchen), betätigen Sie RUN/STOP RESTORE.

Wenn Linien eingezeichnet werden, ohne die gewünschte Form des Dreiecks darzustellen, überprüfen Sie die Befehle DATA für E% und L%. Wenn möglich, bitten Sie jemanden, Ihnen die Programmzeilen aus dem Buch vorzulesen, während Sie sie auf dem Bildschirm überprüfen. Sehen Sie sich Zeile 1520 an und vergewissern Sie sich, ob NL mit 2 und X0 und Y0 mit 0 gleichgesetzt sind. Ist für die FOR/NEXT-Schleife 0 TO 2 eingesetzt? Wenn die Hauptroutine in Ordung ist, schauen Sie sich die Unterroutine an. Bleiben Sie solange am Ball, bis das Dreieck korrekt eingezeichnet ist.

Listen Sie die Zeilen 1500 bis 1530 auf dem Bildschirm. Zeile 1510 enthält den Farbcode für Schwarz vor Blau. Der Bildschirm war zwar schon schwarz auf blau, aber die Kennzeichnung von C ist immer ein guter Einstieg.

Zeile 1520 ist von größter Wichtigkeit. Zuerst bestimmt sie den Wert von NL, welche eine Variable (Platzhalter) für die Anzahl der Linien der zu zeichnenden Form ist. Da die Werte bei 0 beginnen, wird ein Dreieck mit seinen drei Seiten gezeichnet, wenn NL mit 2 gleichgesetzt wird. Die Variable NL wird in der Unterroutine 110 benutzt, so daß sie jedesmal beim Abrufen der Unterroutine bestimmt werden muß. Die Bestimmung von NL bereitet keinerlei Schwierigkeiten, da Sie nichts weiter tun müssen, als NL mit der Nummer der letzten Linie in Ihrer handgeschriebenen Linienliste gleichzusetzen.

Zeile 1520 enthält auch sogenannte "Offset-Werte". Diese X0- und Y0-Werte ermöglichen das Probezeichnen und Verschieben Ihrer Form. Der Wert, den Sie für X0 eingeben, wird einfach zu allen X-Koordinaten in Ihrer Liste der Endpunktdaten addiert. Dies berichtigt die Form, so daß sie rechts oder links von ihrer ursprünglichen Position auf dem Bildschirm eingezeichnet wird. Nehmen wir als Beispiel an, Sie haben dieses Werkzeug benutzt, um eine gerade Linie von Endpunkt 3,3 zu Endpunkt 20,50 einzuzeichnen. Wenn Sie X0 mit 5 gleichsetzen (X0 = 5), addiert der Computer 5 zu jeder X-Koordinate, bevor die Linie eingezeichnet wird. Wenn Sie stattdessen die Linie 5 Spalten nach links verschieben wollen, muß X0 mit -5 gleichgesetzt werden, da 5 von jeder X-Koordinate substrahiert werden muß, um die Form nach links zu verschieben. Denken Sie immer an diese Regel:

-X0 verschiebt die Form nach links; +X0 verschiebt die Form nach rechts

Die Änderung von Y0 verschiebt die Form nach oben oder unten. Wenn eine Linie von 8,3 zu 25,50 eingezeichnet würde, und Sie meinen, sie sollte zwei Reihen weiter nach unten verschoben werden,

würden Sie Y0 mit 2 gleichsetzen (Y0 = 2). So wird 2 zu jeder Y-Koordinate addiert, und die Linie von 8,5 zu 25,52 eingezeichnet (2 Reihen weiter unten auf dem Bildschirm). Für Y0 lautet die Regel:

-Y0 verschiebt die Form nach oben; +Y0 verschiebt die Form nach unten

Zeichnen Sie nun das Dreieck ein, indem Sie es 32 Spalten nach rechts und 32 Reihen nach unten verschieben. Geben Sie ein:

```
1520 NL = 2: X0 = 32: Y0 = 32
```

Starten Sie das Programm wieder. Nachdem der Bildschirm gelöscht ist, wird das Dreieck an der neuen Stelle eingezeichnet (weiter rechts und weiter unten).

Es ist wichtig, daß Sie immer an die möglichen Bereiche für die X- und Y-Koordinaten denken, wenn Sie die Werte für X0 und Y0 bestimmen. Wohin Sie Ihre Form auch verschieben, alle Punkte müssen innerhalb dieser Bereiche bleiben. Wenn Ihre ursprügliche Form eine X-Koordinate mit dem Wert 10 hat und Sie wollen die Form 12 Spalten nach links verschieben (X0 = -12), bekommen Sie Schwierigkeiten. Der Wert der ursprünglichen X-Koordinate, 10, addiert zu -12, ergibt -2. Diese Koordinate liegt außerhalb des zulässigen Bereichs.

Außerdem dürfen Sie nicht vergessen, die Werte von NL, X0 und Y0 für jede neue Form neu zu bestimmen. Dies ist sehr wichtig. X0 und Y0 können auf 0 gesetzt werden (dann wird Ihre Form überhaupt nicht verschoben), aber Sie müssen in jedem Fall bestimmt werden. Wenn Sie diese Variablen bestimmen und die Unterroutine DRAW A SHAPE aufrufen, bleiben die Werte der Variablen solange erhalten, bis Sie jede von ihnen wieder ändern. Benutzt ein Programm die Unterroutine, um mehr als eine Form zu zeichnen, werden alle Formen nach den zuletzt bestimmten Werten von NL, X0 und Y0 gezeichnet.

Vervielfältigen von Formen

Es ist ganz einfach, eine Form zu vervielfältigen. Sie müssen nur neue Offset-Werte bestimmen, einen neuen Farbcode angeben (falls erwünscht) und einen neuen GOSUB 110-Befehl eingeben (einen für jede Kopie der gewünschten Form).

Listen Sie Ihre Hauptroutine auf dem Bildschirm und sehen Sie sich noch einmal an, was Sie bisher eingegeben haben:

- Zeile 1200 reserviert Speicherplatz für zwei Listen
- Zeilen 1300-1340 füllen E% mit einer Beschreibung der Endpunkte des Dreiecks aus
- Zeilen 1400-1440 füllen L% mit einer Beschreibung der Linien des Dreiecks aus
- Zeilen 1500-1530 geben die Anzahl der Linien in der Form an, bestimmen die Offset-Werte und den Farbcode und rufen die Unterroutine DRAW A SHAPE auf

Die Beschreibung des Dreiecks bleibt in E% und L% erhalten, bis Sie die Listen mit der Beschreibung einer neuen Form ausfüllen. Indem Sie neue Offset-Werte eingeben und die Unterroutine wieder aufrufen, kann die Form, so oft Sie wollen, vervielfältigt werden. Geben Sie die folgenden neuen Zeilen in Ihr Programm ein, damit Sie sehen, was wir meinen:

```
1540 NL = 2: X0 = 16: Y0 = 16
1550 GOSUB 110
1560 NL = 2: X0 = 8: Y0 = 8
1570 GOSUB 110
```

Starten Sie das Programm und sehen Sie sich an, was passiert. Es dauert einen Moment, aber am Ende sehen Sie drei Dreiecke - eins für jeden GOSUB 110-Befehl im Programm. Jedes Dreieck wird an einer, im Vergleich zum in E% und L% beschriebenen, ursprünglichen Dreieck, neuen Stelle eingezeichnet. Das anfangs eingegebene Dreieck wird nicht auf dem Bildschirm eingezeichnet. Es ist wichtig, daran zu denken, daß Offset-Werte eine Form nicht von der letzten Kopie der eingezeichneten Form ableiten. Stattdessen leiten Offset-Werte eine Form von der in E% und L% gespeicherten Form ab. Dies wird oft vergessen.

Solange Sie eine Form in E% und L% gespeichert haben, kann das Werkzeug 110 sie zeichnen. Die Variable NL muß mit der Anzahl der Linien in der Form gleichgesetzt werden. Die Numerierung der Linien beginnt immer bei 0. Offset-Werte können benutzt werden, um eine Form zu verschieben oder irgendwo auf dem Bildschirm zu vervielfältigen. Um die Form zu verschieben, setzen Sie X0 mit der Anzahl der Spalten rechts (+) oder links (-) gleich, um die die Form verschoben werden soll. Y0 setzen Sie mit der Anzahl der Reihen darunter (+) oder darüber (-) gleich, um die die Form verschoben werden soll. Dann starten Sie das Programm wieder.

Um eine Form zu vervielfältigen, benötigen Sie den Befehl GOSUB 110 für jede Kopie der gewünschten Form. X0 und Y0 müssen vor jedem GOSUB 110 als Angabe der Position jeder neuen Kopie

geändert werden. NL setzen Sie mit der letzten Liniennummer in der Form gleich.

Nehmen Sie sich etwas Zeit, um mit diesem neuen Werkzeug sicher umgehen zu können. Verschieben Sie das Dreieck und vervielfältigen Sie es über den ganzen Bildschirm. Geben Sie neue Programmzeilen ein, die E% und L% mit der Beschreibung einer neuen Form ausfüllen. Programmzeilen können höchstens zwei Bildschirmzeilen lang sein (80 Zeichen), weshalb DATA-Befehle oft auf mehr als eine Programmzeile verteilt werden müssen. Dazu muß nur jede Zeile mit dem Befehl DATA beginnen.

Wenn Sie mit neuen Programmzeilen eine neue Form in E% und L% speichern, ändern Sie X0 und Y0 wieder zu 0. Andernfalls wird die neue Form um 16 Reihen und Spalten verschoben eingezeichnet (16 ist der zuletzt für X0 und Y0 angegebene Wert).

Der Rest des Kapitels behandelt das Einzeichnen des Schiffes und der Seemöven mit Hilfe des Werkzeugs DRAW A SHAPE. Da das Schiff aus 68 eingezeichneten Linien besteht, lernen Sie eine Menge über dieses neue Werkzeug, bevor Sie ans Ende des Kapitels gelangen. Sie sollten dieses Schiff zuerst einzeichnen, bevor Sie die neue Unterroutine für Ihre eigenen Werke benutzen.

Einzeichnen des Schiffsrumpfes

Drei Dinge sind zu erledigen, bevor Sie das Programm dieses Kapitels eingeben. Als erstes müssen Sie das Bild aus Kapitel 4 in den Speicher laden. Dazu geben Sie ein: LOAD "KAPITEL4.PIC" und LOAD "KAPITEL4.COL". An jeden der Befehle muß die Gerätenummer und ",1" angehängt werden. Lesen Sie im letzten Kapitel noch einmal die entsprechenden Abschnitte nach, wenn Sie Hilfe zum Laden dieser Dateien benötigen. Dann laden Sie das Programm aus Kapitel 4 mit LOAD "KAPITEL4" und der Angabe der Gerätenummer.

Wenn das Bild geladen ist, geben Sie GOSUB 20 ein und betätigen Sie RETURN. Sie sehen das Land, das Wasser, Wellen, den Himmel und den Leuchtturm an gewohnter Stelle. Fahren Sie nicht fort, bis Sie dies tatsächlich auf dem Bildschirm sehen. Dann betätigen Sie RUN/STOP RESTORE und erhalten wieder die Textanzeige.

Sie müssen zur Eingabe des Programms die ZAP-Routine starten, nicht bevor Sie jedoch Zeile 11 folgendermaßen geändert haben:

```
11 A = 256: B = 2049: C = 1003
```

Geben Sie nun RUN 10 ein und betätigen Sie RETURN, worauf die Hauptroutine aus Ihrem Programm gelöscht wird. Dann fügen Sie folgende Programmzeilen hinzu, mit denen Sie den hochauflösenden Bildschirm erhalten und wieder verlassen können:

```
1100 GOSUB 20          : REM GRAPHICS
5000 GET A$
5010 IF A$ = " " THEN 6000
5020 GOTO 5000
6000 GOSUB 30
6010 END
```

Sie merken, daß wir keinen Befehl GOSUB 40 und GOSUB 50 in das Programm eingefügt haben. Diese Unterroutinen löschen jedes Bild auf dem hochauflösenden Bildschirm. Da das Bild aus Kapitel 4 auf dem Bildschirm erhalten bleiben soll, haben wir diese Befehle absichtlich ausgelassen.

Nun können Sie den Schiffsrumpf zeichnen. Geben Sie zuerst den Befehl DIM für die E%- und L%-Listen ein:

```
1200 DIM E%(1,99), L%(1,99)
```

Nachfolgend geben Sie die FOR/NEXT-Schleife ein, die die Endpunkte des Rumpfes liest und in E% einsetzt:

```
1100 REM:::::::::SHIP ENDPOINTS
1110 FOR I = 0 TO 16
1120 READ E%(0,I), E%(1,I)
1130 NEXT I
```

Bevor Sie die folgenden DATA-Befehle eingeben, sehen Sie sich ihre Anordnung an. Alle Kommata stehen exakt untereinander. Wenn Ihnen dies möglicherweise ein bißchen übertrieben vorkommt, erleichtert Ihnen diese Anordung doch erheblich die Suche nach Tippfehlern. Indem Sie den DATA-Befehlen eine einheitliche Länge geben (jeweils mit derselben Anzahl von Dateneinheiten) und die Dateneinheiten genau untereinander setzen, können Sie leicht erkennen, ob Sie eine Zahl hinzugefügt oder ausgelassen haben. Geben Sie die DATA-Befehle genauso ein, wie wir sie angeordnet haben:

```
1140 DATA 114, 108,  93, 125,  89, 125
1150 DATA  93, 135,  84, 150,  78, 150
1160 DATA  56, 132,  54, 135,  92, 127
1170 DATA  90, 130,  18, 144,  17, 141
```

```
1180 DATA   4, 144,   7, 150,   6, 149
1190 DATA  88, 132,  92, 137
```

Überprüfen Sie die Zeilen noch einmal, bevor Sie fortfahren. Eine Gruppe von direkt aufeinanderfolgenden Datenbefehlen wird "Datenblock" genannt. Der obige Datenblock enthält die Endpunkte für den Schiffsrumpf. Diese werden ermittelt, indem Sie den Rumpf auf Millimeterpapier skizzieren und die Koordinaten jedes Endpunktes notieren.

Sie müssen auch die "von"/"zu" Daten für die L%-Liste eingeben:

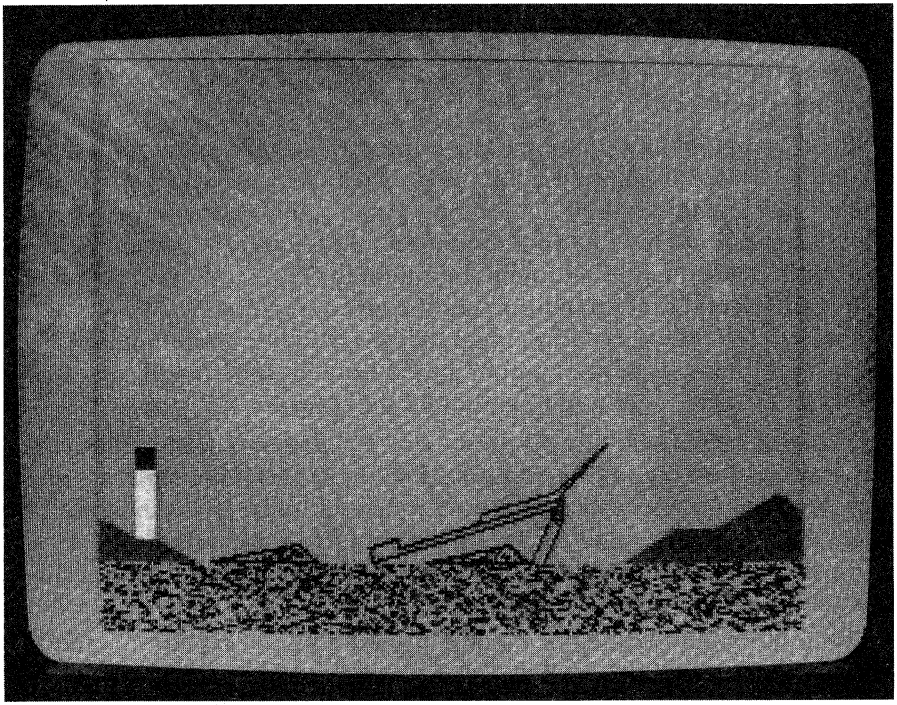
```
1200 REM:::::::::SHIP LINES
1210 FOR I = 0 TO 13
1220 READ L%(0,I), L%(1,I)
1230 NEXT I
1240 DATA  0,  1,  0,  2,  1,  3,  1,  5
1250 DATA  2,  6,  3,  4,  3,  9,  6,  7
1260 DATA  8, 10,  9, 14, 10, 11, 11, 12
1270 DATA 12, 13, 15, 16
```

Überprüfen Sie die DATA-Befehle, bevor Sie fortfahren. Vergewissern Sie sich, daß Zeile 1210 für die Schleife 0 TO 13 angibt.

Zu den Angaben der Endpunkte und der Linien des Schiffes müssen X0 und Y0 bestimmt werden, um das Schiff an die richtige Stelle auf dem Bildschirm zu setzen. Schließlich ist noch ein Befehl GOSUB notwendig, der die benötigte Unterroutine aufruft. Geben Sie folgende Zeilen ein:

```
1300 REM:::::::::DRAW SHIP
1310 C = 14
1320 NL = 13: X0 = 114: Y0 = 26
1330 GOSUB 110
```

Damit sind die Informationen für den Schiffsrumpf vollständig. Sehen Sie sich jede Zeile sorgfältig an, bevor Sie das Programm starten. Wenn möglich lassen Sie noch jemand die DATA-Befehle durchsehen. Dann starten Sie das Programm. Der Rumpf wird folgendermaßen eingezeichnet:



Das Schiff wird in der Mitte über dem Wasser eingezeichnet. Wenn es an einer falschen Stelle erscheint, überprüfen Sie die X0- und Y0-Offsetwerte im Programm. Wenn nur einige Linien an einer falschen Stelle eingezeichnet werden, oder wenn eine Linie überhaupt nicht eingezeichnet wird, überprüfen Sie den Endpunkt-Datenblock in den Zeilen 1140-1180. In diesem Block müssen 34 Zahlen stehen. Schließlich können Sie noch die für L% eingegebenen Daten überprüfen. Sie müssen das Bild aus Kapitel 4 noch einmal laden, bevor Sie ein korrigiertes Programm starten.

Wenn das Programm richtig arbeitet, listen Sie die Zeilen 1020 bis 1320. Da Sie Ihr neues Werkzeug nun benutzt haben, sollen Sie auch mehr über READ-Befehle und FOR/NEXT-Schleifen lernen. Die Kenntnis aller Einzelheiten kann Ihnen eine Menge Verwirrung ersparen, der Sie sich aussetzen, wenn die schönen Entwürfe Ihrer Kunstwerke auf dem Bildschirm nur als Mischmasch erscheinen.

READ veranlaßt den Computer, Daten in eine angegebene Liste einzutragen. Wenn der Befehl lautet "READ L%...", werden Daten in die L%-Liste eingetragen. Lautet der Befehl "READ E%...", werden

Daten in die E%-Liste eingetragen. Woher weiß der Computer, welche Daten er zu lesen hat und wann er aufhören soll zu lesen? Dies klingt wie eine Grundsatzfrage, aber lesen Sie weiter...

Wenn der Computer zu einem READ-Befehl gelangt, setzt er die erste nicht verwendete Dateneinheit, die er findet, in die angegebene Liste ein. So können Sie einen READ-Befehl in Zeile 35400 haben, der einen DATA-Befehl in Zeile 1 liest. Der Computer hört auf, Dateneinheiten zu lesen, wenn die Schleife, in dem der Befehl READ sich befindet, die angegebene Anzahl von Durchgängen ausgeführt hat. Wenn der Computer keine Dateneinheiten mehr findet, die er lesen kann, bevor die Schleife vollendet ist, erscheint eine "OUT OF DATA"-Fehlermeldung und das Programm wird abgebrochen.

Mit anderen Worten, wenn ein Programm läuft, liest ein READ-Befehl Dateneinheiten aus einem DATA-Befehl. Während jede Dateneinheit gelesen und in eine Datenliste eingesetzt wird, wird sie vom Computer "abgehakt". Dies verhindert, daß eine Dateneinheit mehr als einmal gelesen wird. Deshalb ist es so wichtig, daß Sie Ihre Schleifen exakt bestimmen. Wenn Sie beispielsweise die Schleife, die die E%-Daten liest, zu lang bestimmen, werden Dateneinheiten der L%-Liste in die E%-Liste gelesen - ungeachtet ihrer Zugehörigkeit zur L%-Liste. Wenn dann die L%-Liste ausgefüllt wird, sind diese ersten Dateneinheiten schon "abgehakt" und werden entsprechend nicht mehr in diese Liste eingetragen. Die L%-Liste wird zu kurz, ein "OUT OF DATA"-Fehler tritt auf und das Programm wird abgebrochen.

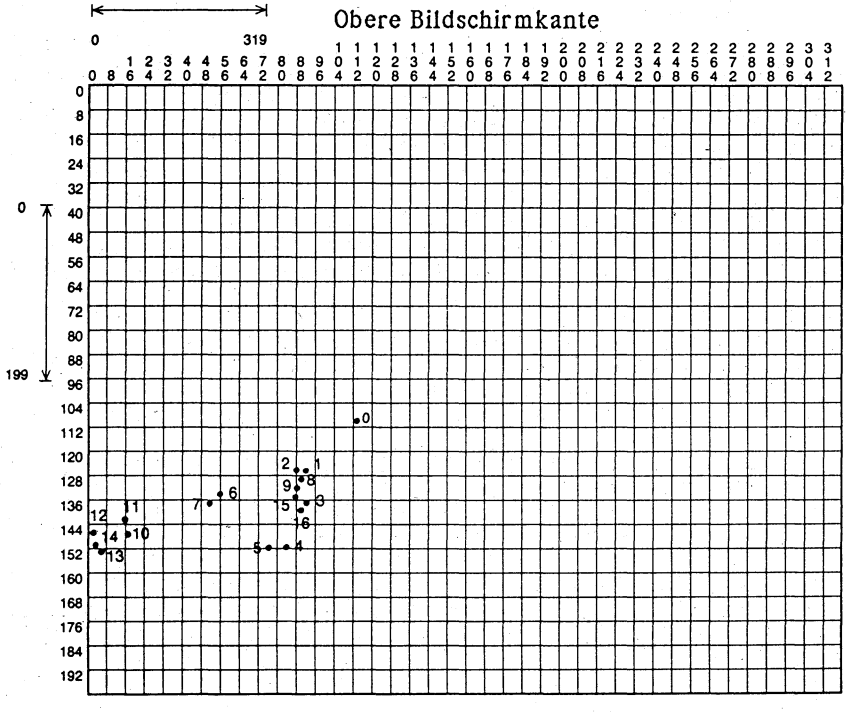
READ-Befehle lesen DATA-Befehle einfach von der kleinsten zur größten Zeilennummer. Wie die DATA-Befehle angeordnet sind, wo sie sich befinden oder in welchem Abschnitt sie erscheinen, hat keinen Einfluß auf die Art und Weise, wie sie gelesen werden. Sobald jede Dateneinheit gelesen und in eine Liste eingetragen ist, wird sie abgehakt - um während der laufenden Programmausführung nicht mehr gelesen zu werden. Allein deshalb sollten Sie genauestens darauf achten, Ihre Schleifen richtig zu bestimmen.

Sehen Sie sich den Datenblock mit den Endpunkten Ihres Schiffes noch einmal an:

```
1140 DATA 114, 108,  93, 125,  89, 125
1150 DATA  93, 135,  84, 150,  78, 150
1160 DATA  56, 132,  54, 135,  92, 127
1170 DATA  90, 130,  18, 144,  17, 141
1180 DATA   4, 144,   7, 150,   6, 149
1190 DATA  88, 132,  92, 137
```

In Programmzeile 1140 ist 114 die X-Koordinate und 108 die Y-Koordinate des Endpunktes #0 Ihres Schiffes. Endpunkt #1 hat die Koordinaten 93,125 (X,Y), Endpunkt #2 die Koordinaten 89,125. Programmzeile 1150 beginnt mit den Koordinaten des Endpunktes #3. Wenn Sie alle Koordinatenpaare zusammenzählen, ergeben sich 17. Deshalb werden für die Schleife 17 Durchgänge (0 TO 16) bestimmt.

Vergleichen Sie die Endpunkte der oben angegebenen DATA-Befehle mit der folgenden Abbildung, in der die Endpunkte Ihres Schiffes auf Millimeterpapier eingezeichnet sind. Sie sehen die Endpunkte an den Stellen, die ursprünglich eingezeichnet worden sind, als das Schiff entworfen wurde. Nach verschiedenen Probezeichnungen wurde das Schiff durch Änderung von X0 und Y0 in Zeile 1310 verschoben.



X, Y Pixelpunkte

In Ihrem Programm wird die Schleife für die Liniendaten auf 14 Durchgänge (0-13) für die 14 Linien des Schiffes festgesetzt. Bei

jedem Durchgang wird ein Satz "von" und "zu" Punkte aus dem DATA-Befehl gelesen und in die L%-Liste eingesetzt.

Das Schiff wird gezeichnet oder vervielfältigt, indem die Unterroutine DRAW A SHAPE mit dem Befehl GOSUB 110 aufgerufen wird.

Werkzeug 110::::::::::Zeichnen einer Form

```
110 REM:::::::::DRAW A SHAPE
111 FOR J = Ø TO NL
112 E1 = L%(Ø,J): E2 = L%(1,J)
113 X1 = E%(Ø,E1) + XØ: Y1 = E%(1,E1) + YØ
114 X2 = E%(Ø,E2) + XØ: Y2 = E%(1,E2) + YØ
115 GOSUB 8Ø
116 NEXT J
117 RETURN
```

Zweck: *Dieses Werkzeug zeichnet eine in den Listen E% und L% beschriebene Form. Es plaziert (oder vervielfältigt) die Form an der durch die Variablen X0 und Y0 angegebenen Stelle.*

Anwendung:

1. *Zeichnen Sie Ihre Form auf Millimeterpapier.*
2. *Numerieren Sie alle Endpunkte Ihrer Form, bei Endpunkt #0 beginnend.*
3. *Stellen Sie eine Liste (E%) der X,Y-Koordinaten jedes Endpunktes zusammen:*

E%

	Endpkt. 0	Endpkt. 1	Endpkt. 2...
X	55	60	55
Y	3	10	15

4. *Stellen Sie eine Liste (L%) der "VON"/"ZU"-Daten zusammen, die für die Beschreibung der Linien Ihres Schiffes benötigt werden:*

L%

	Linie 0	Linie 1	Linie 2...
"VON" Endpkt.#	3	1	2
"ZU" Endpkt.#	2	3	6

5. *Geben Sie einen DIM-Befehl ein, um Platz für die E%- und die L%-Liste zu reservieren:*

```
1020 DIM E%(1,#),L%(1,#)
```

wobei # für die maximale Anzahl der in jeder Liste benötigten Spalten steht.

6. *Füllen Sie E% nach folgendem Schema mit den Endpunktdaten aus:*

```
1110 FOR I = 0 TO #
```

wobei # für die Gesamtzahl der Endpunkte, einschließlich Endpunkt #0, in der zu zeichnenden Form steht.

```
1120 READ E%(0,I),E%(1,I)
1130 NEXT I
1140 DATA X,Y,X,Y
```

wobei X,Y für die Koordinaten jedes Endpunktes steht, bei Endpunkt #0 beginnend.

7. *Füllen Sie L% nach folgendem Schema mit den Liniendaten aus:*

```
1210 FOR I = 0 TO #
```

wobei # für die Gesamtzahl der zu zeichnenden Linien, einschließlich der Linie #0, steht.

```
1220 READ L%(0,I),L%(1,I)1230 NEXT I
1240 DATA von,zu,von,zu
```

wobei "von,zu" für die Endpunktnummern jeder Linie steht, bei Linie #0 beginnend.

8. *Bestimmen Sie die Werte von NL, X0 und Y0:*

```
1310 NL = E: X0 = F: Y0 = G: GOSUB 110
```

E muß die letzte Liniennummer in Ihrer handgeschriebenen Liste der Liniendaten sein; F gibt an, um wieviele Spalten

nach rechts Ihre Form verschoben werden soll, während G die Verschiebung der Form die Reihen hinunter angibt.

Technische Beschreibung: *Um eine Form zu zeichnen, wird eine Schleife benutzt:*

```
111 FOR I = 0 TO NL
```

NL gibt die Anzahl der Linien in der Form an (ebenfalls bei 0 beginnend) und diese Schleife wird einmal für jede Linie durchlaufen. Beim ersten Durchgang wird eine Linie zwischen den Punkten gezeichnet, die an erster Stelle in der L%-Liste eingetragen sind:

```
L%(0,0),L%(1,0)
```

Beim zweiten Durchgang wird die zweite Eintragung in der L%-Liste benutzt:

```
L%(0,1),L%(1,1)
```

Denken Sie daran, daß L%(0,I) und L%(1,I) die beiden Endpunkte sind, die die Linie bestimmen (wobei I von der Schleife durch eine Nummer ersetzt wird). Diese Endpunkte sind vorher von der Hauptroutine in der E%-Liste gespeichert worden.

```
112 E1 = L%(0,I): E2 = L%(1,I)
```

Diese Zeile erhält die Nummern der ersten beiden Endpunkte der Linie, die gerade eingezeichnet werden soll.

```
113 X1 = E%(0,E1) + X0: Y1 = E%(1,E1) + Y0
```

Diese Zeile sucht in E% nach den aktuellen X,Y-Koordinaten des ersten Endpunktes der Linie und addiert dann die Offset-Werte dazu.

```
114 X2 = E%(0,E2) + X0: Y2 = E%(1,E2) + Y0
```

Diese Zeile sucht in E% nach den aktuellen X,Y-Koordinaten des zweiten Endpunktes der Linie und addiert ebenfalls die Offset-Werte dazu.

Wir haben jetzt alle notwendigen Informationen gesammelt, um eine Linie mit der Unterroutine PLOT A LINE in Zeile 80 zu zeichnen.

115 GOSUB 80
116 NEXT I

Das ist das Ende der Schleife.

117 RETURN

Zeile 117 befiehlt dem Computer, zur Hauptroutine zurückzukehren.

Die vorderen Segel

Sie sind nun in der Lage, den Rest weitaus schneller zu bewältigen, da Sie den größten Teil des in diesem Kapitel behandelten Stoffes schon gelernt haben. Die vorderen Segel werden in derselben Weise eingezeichnet, wie der Schiffsrumpf. Geben Sie die folgenden DATA-Befehle für die Endpunkte der Segel ein:

```
1400 REM:::::::::FRONT SAIL ENDPOINTS
1410 FOR I = 0 TO 42
1420 READ E%(0,I), E%(1,I)
1430 NEXT I
1440 DATA  96,   7,  91,  28,  92,  29
1450 DATA  86,  17, 105,  29,  78,  22
1460 DATA 108,  38,  75,  30,  69,  39
1470 DATA  67,  38, 112,  62, 111,  54
1480 DATA 108,  60,  86,  48,  88,  50
1490 DATA  85,  51,  88,  52,  66,  40
1500 DATA 122,  72,  62,  54,  42,  75
1510 DATA  39,  72, 117, 105, 113, 103
1520 DATA 122,  90,  76,  89,  81,  90
1530 DATA  75,  94,  79,  96,  60,  88
1540 DATA 115, 111,  60, 109,  57, 120
1550 DATA  55, 119,  81, 126,  93, 130
1560 DATA 115, 135, 110, 134, 117, 126
1570 DATA  70, 123,  76, 124,  68, 129
1580 DATA  74, 128
```

Nun geben Sie die Programmzeilen ein, die diese Endpunkte verbinden, um die entsprechenden Linien zu bilden:

```
1600 REM:::::::::FRONT SAIL LINES
1610 FOR I = 0 TO 27
1620 READ L%(0,I), L%(1,I)
```

```
1630 NEXT I
1640 DATA  0,  1,  0,  2,  3,  4,  5,  6
1650 DATA  5,  7,  6, 11,  7,  8,  9, 10
1660 DATA 11, 12, 13, 15, 14, 16, 17, 18
1670 DATA 17, 19, 18, 24, 19, 20, 21, 22
1680 DATA 23, 24, 25, 27, 26, 28, 29, 30
1690 DATA 29, 31, 30, 38, 31, 32, 33, 34
1695 DATA 35, 36, 37, 38, 39, 41, 40, 42
```

Zum Schluß bestimmen Sie den Farbcode und die Offset-Werte und rufen die Unterroutine auf:

```
1700 REM:::::::::DRAW FRONT SAIL
1710 C = 14
1720 NL = 42: X0 = 114: Y0 = 26
1730 GOSUB 110
```

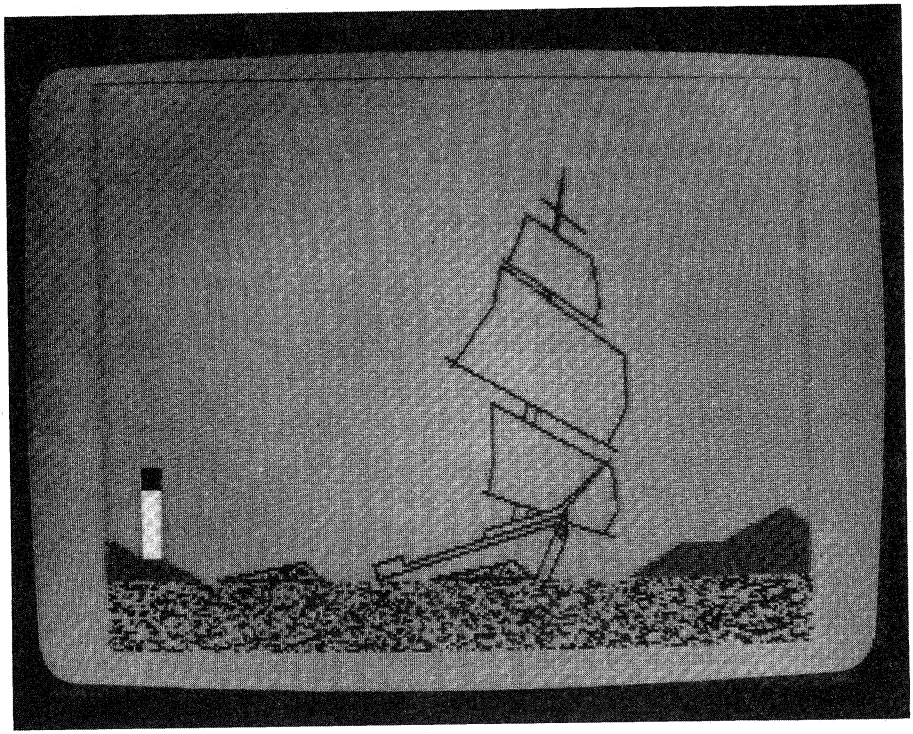
Sie haben vielleicht schon gemerkt, wie leicht Ihnen Fehler in den DATA-Befehlen unterlaufen. (Leider ist, sie aufzuspüren, ein größeres Stück Arbeit.) Wenn ein Programm nicht richtig läuft und eine lange Liste von DATA-Befehlen enthält, wissen Sie jetzt, wo Sie nach Fehlern suchen sollten.

Wenn Sie dieses Programm starten, soll der Computer bei Zeile 1400 beginnen, damit Sie nicht darauf warten müssen, daß der Schiffsrumpf wieder eingezeichnet wird. Leider funktioniert RUN 1400 nicht bei DATA-Befehlen. Warum? Weil ein READ-Befehl (wie der in Zeile 1420) mit dem Lesen der ersten ungelesenen Dateneinheit im Programm beginnt. Diese befindet sich in Zeile 1140 Ihres Programms. Dieser DATA-Befehl wird als erster gelesen, wenn das Programm gestartet wird, ohne Berücksichtigung der Zeilennummer, zu der der Befehl RUN den Computer schickt.

Es gibt aber eine Möglichkeit, dieses Problem zu umgehen. Indem Zeile 1330 in einen REM-Befehl geändert wird (vorübergehend), wird die Form des Schiffsrumpfes in E% und L% gespeichert, aber nicht eingezeichnet. Dann löschen die Zeilen 1400 bis 1695 den Rumpf aus E% und L% und ersetzen ihn durch die Beschreibung der vorderen Segel. Diese werden von Zeile 1730 eingezeichnet. Ändern Sie Zeile 1330 folgendermaßen:

```
1330 REM::GOSUB 110 RETURN
```

Starten Sie das Programm. Die vorderen Segel werden so eingezeichnet, wie Sie sie in der folgenden Abbildung sehen:



Falls Schwierigkeiten auftauchen, gehen Sie nach folgendem Fehlerquellenkatalog vor.

Ist der hochauflösende Bildschirm blockiert, so daß kein Teil der Segel eingezeichnet wird?

- Wenn dies passiert, geben Sie sorgfältig GOSUB 30 RETURN ein. Sie sehen Ihre Eingabe nicht auf dem Bildschirm, müßten aber wieder die Textanzeige erhalten. Sehen Sie eine Fehlermeldung (z.B. "OUT OF DATA ERROR", "BAD SUBSCRIPT ERROR", "SYNTAX ERROR", etc.)?

Werden die vorderen Segel zwar eingezeichnet, scheinen aber willkürlich plaziert zu sein?

- Prüfen Sie, ob die Schleife in Zeile 1210 zu oft durchlaufen wird und deshalb Dateneinheiten aus Zeile 1440 liest (Zeile 1210 muß lauten: FOR I = 0 TO 13).
- Überprüfen Sie die Schleife in Zeile 1410 (FOR I = 0 TO 42). Vergewissern Sie sich, daß sie nicht zu oft durchlaufen wird und zu viele Daten in E% einliest.

- Überprüfen Sie die Liniendaten in den Zeilen 1640-1695. Auslassen oder Hinzufügen von Daten erzeugt von diesem Punkt an ein falsches Bild.

Wurde nur eine Linie falsch eingezeichnet?

- Überprüfen Sie die Datenblöcke der Endpunkte (Zeilen 1440 bis 1580). Eine Nummer wurde wahrscheinlich falsch eingegeben.

Wurde nur eine Linie eingezeichnet?

- Prüfen Sie, ob Sie NL vor GOSUB 110 korrekt bestimmt haben.

Wurden zusätzliche Linien in Ihr Bild eingezeichnet?

- Prüfen Sie, ob Sie NL vor GOSUB 110 korrekt bestimmt haben.

Wurden die vorderen Segel eingezeichnet, befinden sich aber zu weit rechts oder links?

- Überprüfen Sie die Offset-Werte in Zeile 1720. Für X0 muß 14, für Y0 26 eingesetzt werden.

Falsche Farben?

- Überprüfen Sie Zeile 1710.

Wenn Sie Programmzeilen falsch eingegeben haben, müssen Sie:

1. diese korrigieren;
2. das Bild aus Kapitel 4 neu laden;
3. das Programm aus Kapitel 5 neu starten.

Wenn das Programm richtig läuft und das Bild mit dem Schiffsrumpf und den vorderen Segeln erscheint, betätigen Sie die LEERTASTE, um wieder die Textanzeige zu erhalten.

Die hinteren Segel

Geben Sie nun sorgfältig die folgenden Endpunktdaten für die hinteren Segel ein:

```
1800 REM:::::::::REAR SAIL ENDPOINTS
1810 FOR I = 0 TO 40
1820 READ E%(0,I), E%(1,I)
1830 NEXT I
1840 DATA  78,  12,  75,  20,  73,  15
1850 DATA  80,  20,  61,  13,  78,  22
1860 DATA  60,  18,  48,  25,  44,  22
1870 DATA  69,  39,  52,  30,  50,  38
1880 DATA  39,  45,  37,  44,  60,  56
1890 DATA  99,  59,  99,  55,  59,  57
1900 DATA  37,  48,  33,  69,  20,  84
1910 DATA  13,  80,  60, 102, 102, 106
1920 DATA 100,  99,  28,  87,  30,  96
1930 DATA  24, 105,  21, 102,  58, 116
```

```
1940 DATA  15,  81,  25, 141,  41, 111
1950 DATA  49, 114,  33, 140,  42, 138
1960 DATA  54, 115,  39, 139,  48, 137
1970 DATA  12, 105,  63, 127
```

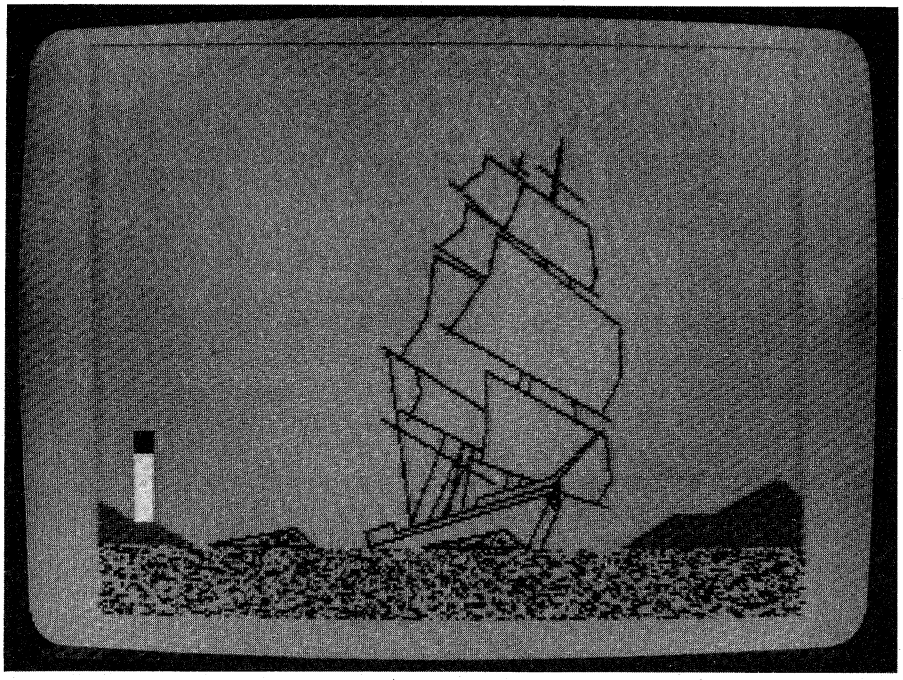
Bevor Sie fortfahren, überprüfen Sie Ihre Eingabe. Dann geben Sie die Liniendaten mit folgenden Zeilen ein:

```
2000 REM:::::::::REAR SAIL LINES
2010 FOR I = 0 TO 25
2020 READ L%(0,I), L%(1,I)
2030 NEXT I
2040 DATA  0,  1,  2,  3,  4,  5,  4,  6
2050 DATA  6,  7,  8,  9,  9, 10, 10, 11
2060 DATA 11, 12, 13, 14, 15, 16, 17, 18
2070 DATA 18, 19, 19, 20, 21, 22, 23, 24
2080 DATA 25, 26, 26, 27, 28, 29, 30, 31
2090 DATA 31, 32, 33, 34, 33, 35, 36, 37
2095 DATA 36, 38, 39, 40
2100 REM:::::::::DRAW REAR SAIL
2110 C = 14
2120 NL = 25: X0 = 114: Y0 = 26
2130 GOSUB 110
```

Sehen Sie sich die Zeilen noch einmal an. Wenn Sie meinen, daß sie korrekt eingegeben sind, ändern Sie Zeile 1730 in einen REM-Befehl, so daß das Programm die vorderen Segel nicht noch einmal einzeichnet. Geben Sie ein:

```
1730 REM::GOSUB 110 RETURN
```

Starten Sie das Programm. Es dauert einen Moment, die Segel einzuzeichnen, ruhen Sie sich also etwas aus. Das Bild muß folgendermaßen aussehen:



Wenn irgendwelche Schwierigkeiten auftauchen, gehen Sie nach der oben angegebenen Checkliste auf Fehlersuche.

Zeichnen der großen Seemöven

In diesem Abschnitt setzen Sie mit Hilfe der Fähigkeit Ihres Werkzeuges 110, Formen zu vervielfältigen, drei große Seemöven in den Himmel. Im nächsten Abschnitt werden auf demselben Weg zwei kleine Seemöven in den Himmel gesetzt. Vögel verschiedener Größen bringen Abwechslung in Ihr Bild und erzeugen eine perspektivische Illusion.

Geben Sie diese Programmzeilen für die großen Möven ein:

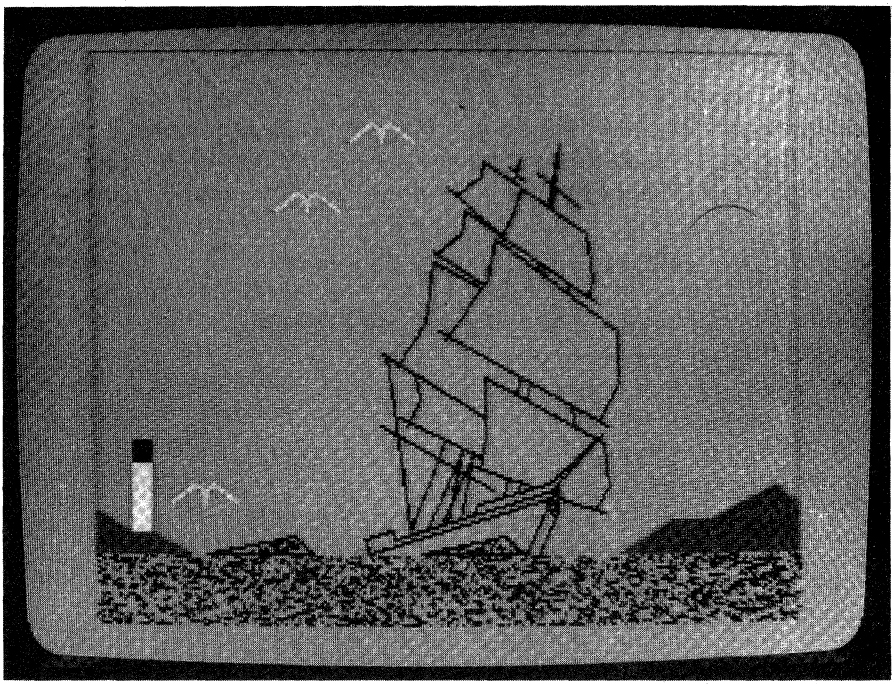
```
2200 REM:::::::::LRG SEAGULL ENDPTS
2210 FOR I = 0 TO 6
2220 READ E%(0,I), E%(1,I)
2230 NEXT I
2240 DATA  0,  6, 10,  0, 13,  2, 14
2250 DATA  6, 15,  2, 18,  0, 28,  6
```

```
2300 REM:::::::::LRG SEAGULL LINES
2310 FOR I = 0 TO 5
2320 READ L%(0,I), L%(1,I)
2330 NEXT I
2340 DATA 0, 1, 1, 2, 2, 3, 3, 4
2350 DATA 4, 5, 5, 6
2400 REM:::::::::DRAW LRG SEAGULLS
2410 C= 30: REM COLOR = WHITE ON BLUE
2420 NL = 5: X0 = 114: Y0 = 26: GOSUB 110
2430 NL = 5: X0 = 33: Y0 = 151: GOSUB 110
2440 NL = 5: X0 = 80: Y0 = 50: GOSUB 110
```

Überprüfen und korrigieren Sie, falls erforderlich, Ihre Eingabe und ändern Sie die Programmzeile 2130 in folgenden REM-Befehl:

```
2130 REM::GOSUB 110 RETURN
```

Dann starten Sie das Programm. Auf Ihrem Bild erscheinen die großen Seemöven wie in der folgenden Abbildung:



Mit der LEERTASTE erhalten Sie wieder die Textanzeige. Listen Sie die Zeilen 2200-2430. Sie sehen, daß die Endpunkt- und Liniendaten nur einmal für alle drei Vögel eingegeben worden sind. Um denselben Vogel dreimal zu zeichnen, wurden neue Offset-Werte eingegeben, jedesmal gefolgt von dem Befehl GOSUB 110.

Bevor Sie fortfahren, ändern Sie die Zeilen 2420, 2430 und 2440 in REM-Befehle:

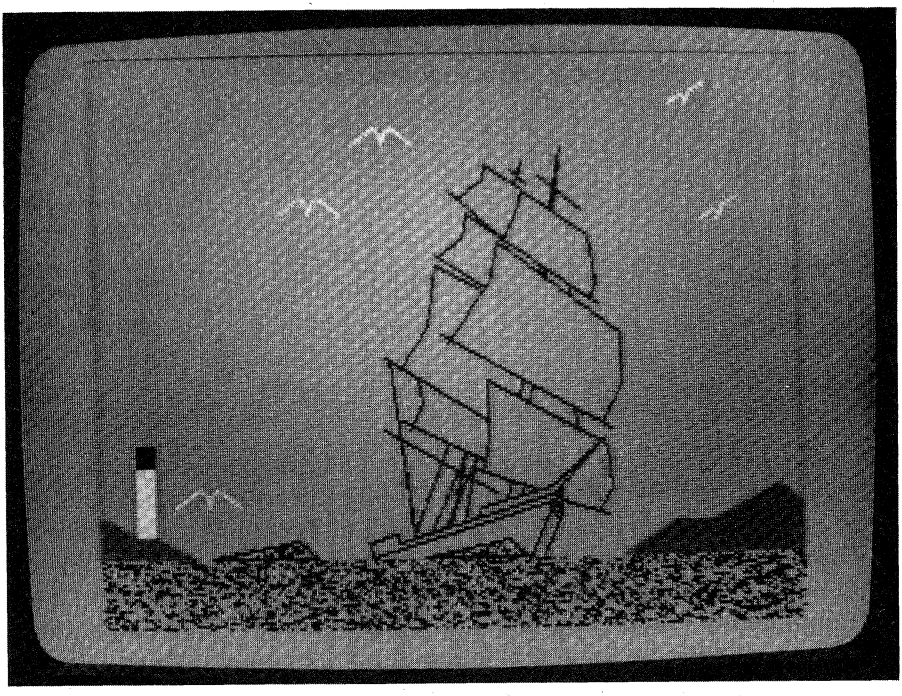
```
2420 REM NL = 5: X0 = 114: Y0 = 26: GOSUB 110
2430 REM NL = 5: X0 = 33: Y0 = 151: GOSUB 110
2435 REM NL = 5: X0 = 80: Y0 = 50: GOSUB 110
```

Vervielfältigen der kleinen Seemöven

Die kleinen Seemöven werden in derselben Weise gezeichnet wie die großen Vögel, indem die Listen E% und L% ausgefüllt werden. Geben Sie die folgenden Zeilen ein:

```
2500 REM:::::::::SM SEAGULL ENDPTS
2510 FOR I = 0 TO 5
2520 READ E%(0,I), E%(1,I)
2530 NEXT I
2540 DATA 0, 8, 5, 5, 8, 6, 7, 8, 9, 4, 16, 0
2600 REM:::::::::SM SEAGULL LINES
2610 FOR I = 0 TO 3
2620 READ L%(0,I), L%(1,I)
2630 NEXT I
2640 DATA 0, 1, 1, 2, 3, 4, 4, 5
2700 REM:::::::::DRAW SM SEAGULLS
2710 NL = 3: X0 = 261: Y0 = 10: GOSUB 110
2720 NL = 3: X0 = 275: Y0 = 50: GOSUB 110
```

Überprüfen Sie Ihre Eingabe und starten Sie das Programm. Mit den beiden kleinen Seemöven haben Sie das Programm dieses Kapitels vervollständigt. Ihr Bildschirm muß nun folgendermaßen aussehen:



Bevor Sie das Programm sichern, löschen Sie alle REMs, die Sie in die Befehle GOSUB 110 eingefügt haben. Die Befehle GOSUB 110 bleiben erhalten, Sie löschen nur das Wort REM. Folgende Zeilen müssen geändert werden:

1330

1730

2130

2420

2430

2440

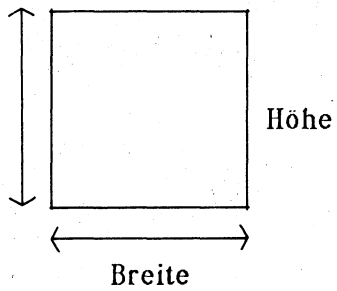
Dann speichern Sie dieses Programm unter dem Dateinamen "KAPITEL5" ab. Wenn es sich sicher auf Diskette/Band befindet,

speichern Sie das Bild unter "KAPITEL5" ab (Informationen zum Sichern von Bildern finden Sie in Kapitel 4).

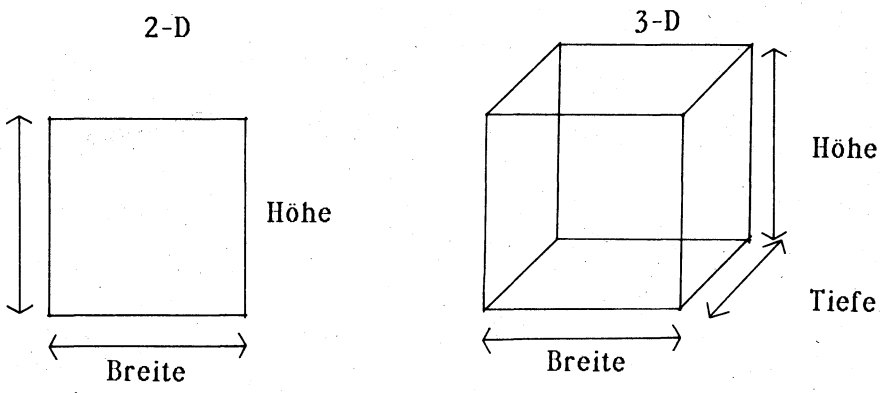
Anregungen für eigene Entwürfe

Mit Ihrem neuen Werkzeug können viele komplexe Konstruktionen einfach in Ihr Bild programmiert werden. Dieser Abschnitt behandelt einige Zeichentechniken, die speziell mit dem Werkzeug DRAW A SHAPE gut ausgeführt werden können. Am Ende des Abschnitts finden Sie die Zusammenfassung dieses Kapitels und zwei Übungsaufgaben.

Eine Form, wie die unten abgebildete, hat eine Höhe (vom kleinsten zum größten Y-Wert) und eine Breite (vom kleinsten zum größten X-Wert), die auch als "Dimensionen" bezeichnet werden. Sind ausschließlich Höhe und Breite Bestandteile einer Form, nennt man sie "zweidimensional. Auf dem **Commodore 64** gezeichnete Formen sind zweidimensional.

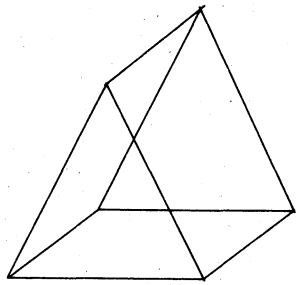


Formen haben in der Realität jedoch eine zusätzliche Dimension, die der Tiefe, weshalb sie als "dreidimensional" bezeichnet werden. Wir leben in einer Welt dreidimensionaler Formen. Es gibt unter den Dingen, die wir berühren können, nichts, das nicht Höhe, Breite und Tiefe hat. Bildern ohne die dritte Dimension der Tiefe fehlt die Wirklichkeitsnähe, die viele Künstler darzustellen versuchen. Es gibt aber eine Möglichkeit, die Illusion der Dreidimensionalität über die Perspektive zu erzeugen. Dies erreichen Sie durch Vervielfältigen einer Form und Verbinden einiger Linien. Das einfache, ebene Quadrat, das Sie oben sehen, kann dreidimensional (3-D) erscheinen, indem Sie eine Kopie weiter oben und rechts auf dem Bildschirm anfertigen und die Ecken des Originals und der Kopie verbinden:

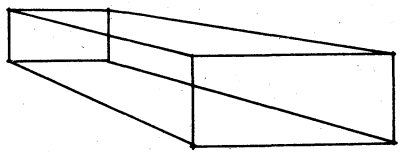


Sie sehen, daß der dreidimensionale Würfel aus zwei Quadraten gebildet ist, die beide identisch sind und unter Zuhilfenahme derselben E%- und L%-Listen gezeichnet werden.

Sie können auch ein dreidimensionales Dreieck mit Hilfe von Offset-Werten zeichnen:



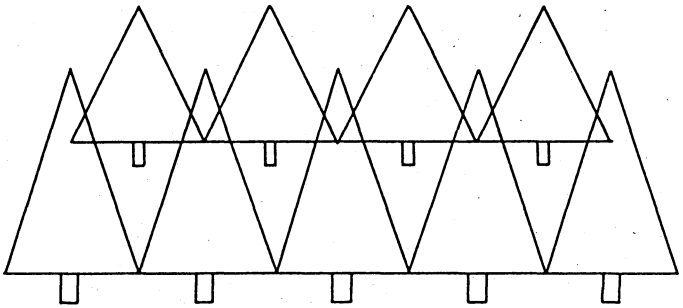
Auf ähnliche Weise kann die folgende Form gezeichnet werden:



In dieser Form ist das vordere Rechteck größer als das hintere. Diese Änderung der Größenverhältnisse entspricht eher der Wirklichkeit, da alle Gegenstände kleiner aussehen, je weiter sie entfernt sind.

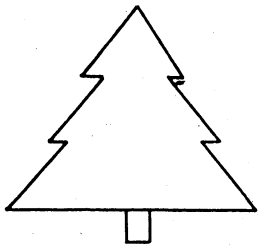
Beachten Sie: Die Unterroutine DRAW A SHAPE kann eine Form nicht verkleinern. Sie müssen sowohl die große als auch die kleine Form zeichnen, um dieses Bild zu zeichnen.

Formen, die sich weiter oben auf dem Bildschirm befinden, scheinen ebenfalls weiter entfernt zu sein. Sie können diese Zeichenmöglichkeit benutzen, um einen Wald zu zeichnen, für den zwei verschieden große Bäume auf dem Bildschirm vervielfältigt werden:

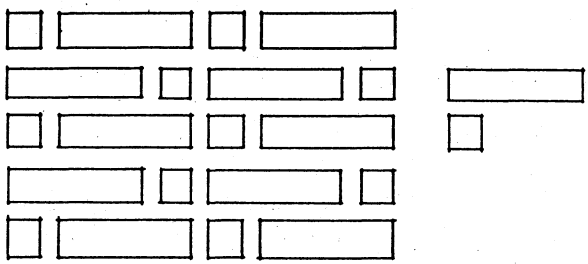


In diesem Beispiel sind die Bäume in der oberen Reihe kleiner als die in der unteren Reihe. Die Veränderung der Größen und Positionen von Gegenständen auf dem Bildschirm erzeugt eine größere Tiefenwirkung.

Interessante Entwürfe entstehen durch unterschiedliche Arten von Formen, unterschiedliche Plazierungen und unterschiedliche Farben. Diese Vielfalt verbessert das Aussehen eines Bildes. Baumformen können durch Hinzufügen weiterer Details interessanter gestaltet werden. Je detaillierter, desto eher erinnern Formen an bestimmte Dinge. Untenstehend z.Bsp. die Umrisse einer Tanne:



Mit der Unterroutine DRAW A SHAPE können Sie schnell einen Tannenwald zeichnen. Diese Unterroutine erweist sich als ebenso nützlich bei der Erstellung von Zeichenmustern, z.B. einer Ziegelwand. Diese besteht aus Rechtecken, die vervielfältigt ein Muster ergeben. Das folgende Muster wurde aus zwei verschieden großen Ziegeln gebildet:



Durch Vervielfältigung von Formen können Sie interessante Entwürfe und farbige Muster erstellen. Mit unterschiedlichen Farben, Größen und Plazierungen einer rechteckigen Form können Sie eine moderne Stadt voller Gebäude entwerfen. Mit einer Kreisform können Sie den Weltraum mit Planeten darstellen. Mit einer Sammlung von Formen können Sie einen Blumenstrauß bilden.

Wie Sie sehen, eignet sich das Werkzeug DRAW A SHAPE hervorragend, um alle Arten von Mustern, Formen und Konstruktionen zu bilden. Durch Üben und Experimentieren sind Sie bald in der Lage, alle Arten interessanter und vielfältiger Entwürfe zu entdecken und umzusetzen.

Zusammenfassung

Die letzten beiden Kapitel waren bis jetzt die schwierigsten. Nun wollen Sie vielleicht endlich Ihre eigenen Ideen auf den Bildschirm bringen. Nehmen Sie sich aber noch kein größeres Werk vor, bis Sie auch das letzte Kapitel dieses Buches gelesen haben - es lohnt die Mühe.

Die Unterroutine DRAW A SHAPE ist das Werkzeug, das in diesem Kapitel zu Ihrem Werkzeugkasten hinzugekommen ist. Obwohl Sie wissen, wie Sie es benutzen können, sind Sie vielleicht der Meinung, daß der Nutzen dieses Werkzeugs den Aufwand nicht wert ist. Schließlich kann das Werkzeug PLOT A LINE Formen zeichnen und ist einfacher zu benutzen. Richtig? Jein. Abhängig von der zu

zeichnenden Form und von der Häufigkeit ihres Vorkommens in Ihrem Bild kann die Unterroutine DRAW A SHAPE Ihnen eine Menge Zeit und Arbeitsaufwand ersparen. Wenn Sie dieses Werkzeug als ein unnötiges "Extra" beiseite legen, verlieren Sie zwei sehr nützliche Funktionen: Probezeichnen und Vervielfältigen von Formen.

Folgende Punkte müssen Sie bei der Benutzung Ihres neuen Werkzeugs immer bedenken:

1. Für die Schleife I muß die korrekte Anzahl der Durchläufe bestimmt werden. Für die E%-Liste muß I von 0 bis zur Nummer des letzen Endpunktes in der Liste durchlaufen werden. Für die L%-Liste muß I von 0 bis zur Nummer der letzten Linie durchlaufen werden.
2. Dateneinheiten in den DATA-Befehlen werden "abgehakt", wenn sie in eine Liste eingelesen werden. Vergewissern Sie sich, daß keine Daten auf Grund einer falsch bestimmten Schleife vorzeitig eingelesen werden.
3. Um die Unterroutine aufzurufen, müssen X0, Y0 und NL bestimmt werden.

Weitere Einzelheiten zur Benutzung dieses Werkzeugs finden Sie in der "Technischen Beschreibung" der Unterroutine. Lesen Sie diesen Abschnitt, wenn Sie Informationen benötigen.

Kapitel 6 beschließt das Buch mit vollständigen Instruktionen zur Herstellung und Steuerung von "Sprites". Ein Sprite ist ein kleines Objekt oder eine Form, die sich auf dem Bildschirm bewegt. Diese Sprites bringen Leben in Ihre Bilder und bilden ein unterhaltsames Finale Ihrer Grafikübungen.

Unten finden Sie wieder zwei Übungsaufgaben, mit denen Sie Ihre Fähigkeiten im Umgang mit dem Werkzeug DRAW A SHAPE testen können. Lösen Sie beide, bevor Sie zu Kapitel 6 übergehen.

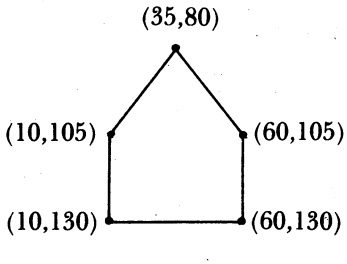
Aufgabe 1

Starten Sie die ZAP-Routine (geben Sie RUN 10 RETURN ein). Dann geben Sie diese neuen Programmzeilen ein, um den Bildschirm zu löschen und zum Einzeichnen vorzubereiten:

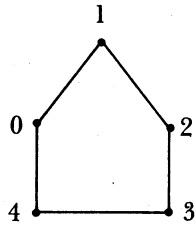
```
1010 GOSUB 20: REM GRAPHICS
1020 C = 14: GOSUB 40: REM COLORS
1030 GOSUB 50: REM PAINT BKGD
```

Ihre erste Aufgabe besteht darin, die Programmzeilen einzugeben, die die unten abgebildete Form zeichnen. Zeichnen Sie sie mit Hilfe Ihres Werkzeugs DRAW A SHAPE und der Listen E% und L%. Sie

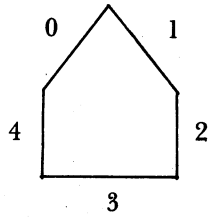
sehen die X,Y-Koordinaten, die Endpunktnummern und die Liniennummern in der Abbildung angegeben. Beginnen Sie bei Programmzeile 1100 und setzen Sie alle Offset-Werte auf 0.



Koordinaten



Endpunkt#'S



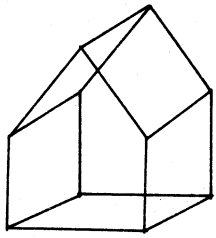
Linie#'S

Lösung 1

```
1100 DIM E%(1,99), L%(1,99)
1200 REM:::::::::HOUSE ENDPOINTS
1210 FOR I = 0 TO 4
1220 READ E%(0,I), E%(1,I)
1230 NEXT I
1240 DATA  10, 105,  35,  80,  60, 105
1250 DATA  60, 130,  10, 130
1260 REM:::::::::HOUSE LINES
1270 FOR I = 0 TO 4
1280 READ L%(0,I), L%(1,I)
1290 NEXT I
1300 DATA 0, 1, 1, 2, 2, 3, 3, 4, 4, 0
1310 REM:::::::::DRAW HOUSE
1320 C = 14
1330 NL = 4: X0 = 0: Y0 = 0
1340 GOSUB 110
```

Aufgabe 2

Lassen Sie Ihr Haus dreidimensional auf dem Bildschirm erscheinen, indem Sie eine Kopie des Originals 25 Spalten weiter rechts und 10 Reihen weiter oben einzeichnen. Dann verbinden Sie die Endpunkte jeder Ecke wie in der folgenden Abbildung:



Tip: Die Linien, die die Eckpunkte des Originals und der Kopie verbinden, lassen sich ganz einfach mit dem Werkzeug PLOT A LINE einzeichnen. Um die X,Y-Koordinaten der Endpunkte in der Kopie zu berechnen, addieren Sie 25 zu jedem X-Wert der Originalkopie und subtrahieren 10 von jedem Y-Wert des ursprünglichen Entwurfs.

Lösung 2

```
1410 NL = 4: X0 = 25: Y0 = -10: GOSUB 110
1420 REM:::::::::DRAW LINES
1430 X1 = 10: Y1 = 130
1440 X2 = 35: Y2 = 120: GOSUB 80
1450 X1 = 10: Y1 = 105
1460 X2 = 35: Y2 = 95: GOSUB 80
1470 X1 = 35: Y1 = 80
1480 X2 = 60: Y2 = 70: GOSUB 80
1490 X1 = 60: Y1 = 105
1500 X2 = 85: Y2 = 95: GOSUB 80
1510 X1 = 60: Y1 = 130
1520 X2 = 85: Y2 = 120: GOSUB 80
```

Kapitel 6
Anfertigen und Steuern von Sprites

Dieses Kapitel behandelt Sprite-Grafik, eine der interessantesten und doch enorm einfach zu benutzenden Grafiktechniken. Sprites sind kleine eingezeichnete Objekte oder trickbildähnliche Figuren, die auf dem Bildschirm bewegt werden können. Sie sind deshalb so interessant, weil sie Leben in Ihre Bilder bringen. Sie sind leicht zu erzeugen und zu steuern, da das meiste durch GOSUB-Befehle erledigt wird. Dieses Kapitel führt Sie schrittweise durch den Herstellungsprozeß und die Kontrolle über die Bewegung eines Sonnen-Sprites auf dem blauen Himmel in Ihrem Bild.

Zuerst muß der hochauflösende Bildschirm das Schiff, die Segel, das Land, das Wasser, die Wellen, die Möven und den Leuchtturm anzeigen. Wenn Sie dies schon auf dem Bildschirm sehen, gehen Sie zum nächsten Abschnitt über. Andernfalls laden Sie das Programm aus Kapitel 5 ("KAPITEL5"), das Bildmuster ("KAPITEL5.PIC") und die Bildfarben ("KAPITEL5.COL"). Um das Bild zu sehen, geben Sie GOSUB 20 ein und betätigen RETURN. Die Textanzeige erhalten Sie mit RUN/STOP RESTORE zurück. Fahren Sie erst mit dem nächsten Abschnitt fort, wenn das Bild richtig auf den hochauflösenden Bildschirm geladen wurde.

Wenn Sie den "Programmer's Reference Guide" oder den "User's Guide" zum **Commodore 64** besitzen, haben Sie vielleicht schon das Kapitel über Sprites gelesen. In diesen Büchern wird ein Sprite in Form eines Heißluftballons durch Starten des entsprechenden BASIC-Programms auf dem Bildschirm bewegt. Im nächsten Abschnitt lernen Sie nicht nur, was ein Sprite ist und wie Sie eines erstellen, sondern auch alles über die interessanten Eigenschaften der Sprites.

Einführung in die Sprites

Was ist ein Sprite?

Ein Sprite gleicht den kleinen beweglichen Figuren in Videospielen. Es ist eine kleine Form, die auf dem Bildschirm bewegt werden kann, um ein lebendiges Bild, ähnlich einem Trickfilm zu erzeugen. "Lebendig" bedeutet hier nichts weiter, als daß sich im Bild etwas

bewegt. Die Beweglichkeit der Sprites unterscheidet diese von allen anderen Formen, die Sie schon eingezeichnet haben.

Stellen Sie sich ein Sprite als eine kleine, herausgelöste Figur vor, die unabhängig von den schon vorhandenen Figuren auf dem Bildschirm bewegt werden kann. Sie kann sich nach oben, unten, rechts, links und diagonal bewegen und sich hinter oder vor anderen Objekten auf dem Bildschirm bewegen. Sie kann vom Bildschirm verschwinden und dort erscheinen. Sie kann sich sogar in einen Farbenblock bewegen, ohne daß dieser Teil des Bildes sich dadurch verändert. Jedes Sprite ist eine getrennte, eigenständige Abbildung, die sich unabhängig von anderen eingezeichneten Formen, Linien, Punkten oder sogar anderen Sprites verhält.

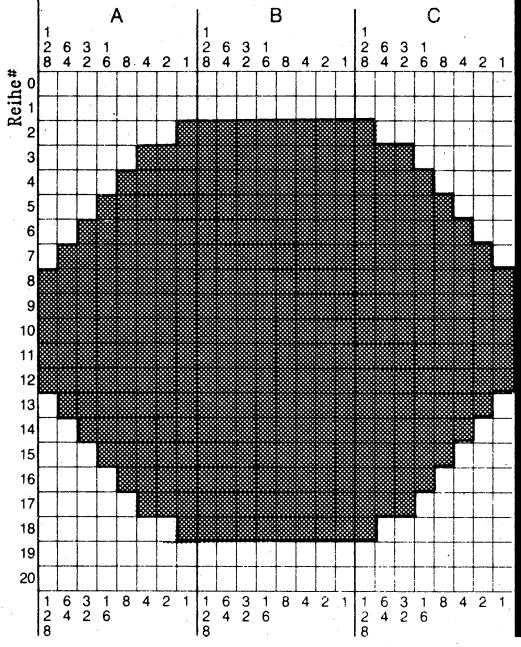
Sie können Ihre Sprites entwerfen, zeichnen, vergrößern und bewegen. In diesem Kapitel werden Sie ein Sprite in Form einer gelben Sonne über den Himmel in Ihrem Bild bewegen. Alle besonderen Merkmale der Sprites werden im Detail behandelt. Der nächste Abschnitt behandelt die verschiedenen Stufen beim Entwurf von Sprites. Wenn Sie das Sprite aus dem "User's Guide" schon kennen, werden Sie sehen, daß die Sonne in unserem Bild in derselben Weise gebildet wird.

Entwurf eines Sprites

Der Entwurf eines Sprites ähnelt dem Zeichnen und Konstruieren Ihres Hauptbildes. Dies geschieht in einem Raster (Millimeterpapier), in dem jedes kleine Quadrat ein Pixel darstellt. Sprites müssen in einem Block von 504 Pixels erstellt werden. Dieser Pixelblock, genannt "Sprite Entwurfsraster", ist 24 Pixels breit und 21 Pixels hoch (24x21=504 Pixels). Die Abbildung des Sprites - sein späteres Aussehen - wird innerhalb dieses Rasters bestimmt. Sehen Sie sich das Raster an, in dem unsere Sonne schon skizziert ist.

Sprite Entwurfsraster

(oben)



DATA - Befehle

BASIC-Zeile#	DATA	Summe aus A	Summe aus B	Summe aus C
1610	DATA	0	0	0
1620	DATA	0	0	0
1630	DATA	1	255	128
1640	DATA	7	255	224
1650	DATA	15	255	240
1660	DATA	31	255	248
1670	DATA	63	255	252
1680	DATA	127	255	254
1690	DATA	255	255	255
1700	DATA	255	255	255
1710	DATA	255	255	255
1720	DATA	255	255	255
1730	DATA	255	255	255
1740	DATA	127	255	254
1750	DATA	63	255	252
1760	DATA	31	255	248
1770	DATA	15	255	240
1780	DATA	7	255	224
1790	DATA	1	255	128
1800	DATA	0	0	0
1810	DATA	0	0	0

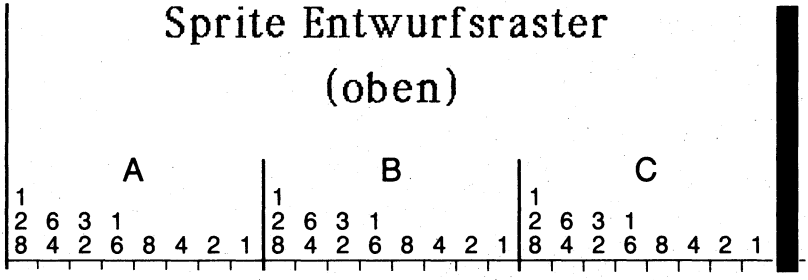
Ein Sprite wird zuerst auf solch einem Entwurfsraster skizziert. (Im Anhang finden Sie ein leeres Entwurfsraster, das Sie kopieren können, um Ihre eigenen Sprites zu entwerfen.) Die erste Stufe beim Entwurf eines Sprites bildet das Skizzieren eines Umrisses mit Bleistift auf dem Entwurfsraster. Zeichnen Sie den Umriß nur schwach ein, damit das Raster noch zu sehen ist. Dann schattieren Sie, ebenfalls nur leicht, die Quadrate innnerhalb des Umrisses. Es ist einfacher, zuerst die Umrisse der Form zu zeichnen und dann die Quadrate zu schraffieren. Zwei wichtige Regeln müssen Sie beim Entwurf eines Sprites immer beachten:

1. Jedes Quadrat des Rasters stellt ein Pixel auf dem Bildschirm dar, deshalb darf Ihr Entwurf kein Quadrat zerteilen. Sie müssen immer das ganze Quadrat schattieren.
2. Ein Sprite kann immer nur einfarbig sein. Die Sonne beispielsweise ist einfarbig gelb.

Wenn Sie ein Sprite skizziert und schattiert haben, müssen Sie die DATA-Befehle sammeln, die das Sprite beschreiben. Dies ist der einzige Vorgang bei der Erstellung und Bewegung von Sprites, der wirk-

lich Konzentration erfordert. Lesen Sie die nächsten Abschnitte deshalb sehr aufmerksam.

Schauen Sie sich den oberen Teil des Entwurfsrasters in der folgenden Abbildung an. Das Raster ist senkrecht in drei Abschnitte unterteilt. Diese Abschnitte sind mit A, B und C bezeichnet. Jeder Abschnitt enthält 8 Pixelspalten. In jedem Abschnitt sind die Spalten numeriert: 128, 64, 32, 16, 8, 4, 2 und 1.



Anhand dieser Nummern werden die notwendigen DATA-Befehle gesammelt. Jede Nummer stellt den Wert dar, der jedem schattierten Pixel in ihrer Spalte zugewiesen ist. Also hat jedes schattierte Pixel in der ersten Spalte einen Wert von 128. Sehen Sie sich noch einmal den Entwurf für die Sonne an. In der ersten Spalte wären die Pixels der Reihen 8 bis 12 betroffen. Jedes schattierte Pixel in der zweiten Spalte hat einen Wert von 64, die schattierten Pixels der dritten Spalte haben einen Wert von 32 usw. Sie haben sicher bemerkt, daß die Betonung in der obigen Ausführung auf "jedes" und "schattierte" lag. Wir sprechen nicht davon, daß eine ganze Pixelspalte einen Wert von 128, 64 oder 32 hat, sondern daß nur die schattierten Pixels betroffen sind. Ist ein Quadrat (Pixel) nicht schattiert, hat es einen Wert von Null (0), da es nicht Teil des Sprites ist. Sie werden nicht in der Lage sein, etwas über Sprites zu lernen, wenn Sie die Werte, die mit jedem Pixel in Ihrem Entwurf verbunden sind, nicht verstehen.

Für jede Reihe Ihres Rasters müssen Sie drei Summen aus diesen Werten errechnen. Zuerst errechnen Sie die Summe der Werte für die 8 Quadrate in Abschnitt A, dann die für die 8 Quadrate in Abschnitt B und schließlich die für die 8 Quadrate in Abschnitt C. Diese drei Summen werden auf der rechten Seite jeder Reihe unter SUMME AUS A, SUMME AUS B und SUMME AUS C notiert.

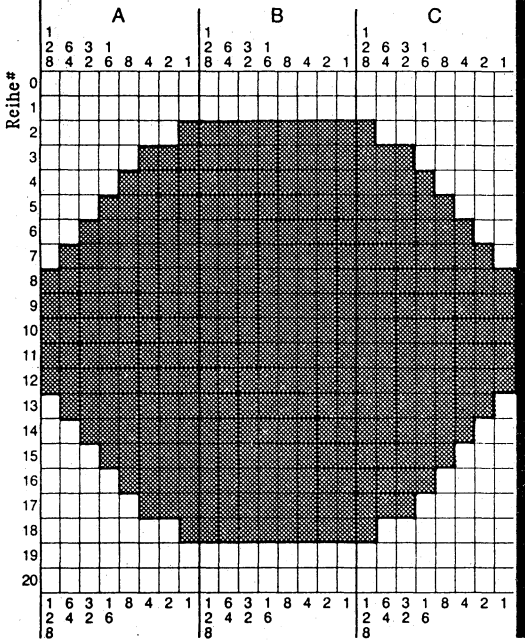
Diese "Summen" sind Ihre Daten. Jede horizontale Reihe in Ihrem Raster entspricht drei getrennten Dateneinheiten, die der Computer

lesen kann. Der Computer kann dann das Sprite aus diesen Datensummen definieren. Anstatt 504 verschiedene Zahlen für 504 einzelne Pixels anzugeben, müssen Sie nur 63 Zahlen (21 Reihen x 3 Datensummen = 63) bestimmen. Später werden diese Datensummen als DATA-Befehle in das Programm eingegeben.

Anhand des Entwurfs zeigen wir Ihnen nun, wie die 63 Datensummen ermittelt werden.

Sprite Entwurfsraster

(oben)



DATA - Befehle

BASIC-Zeile#	DATA	Summe aus A	Summe aus B	Summe aus C
1610	DATA	0	0	0
1620	DATA	0	0	0
1630	DATA	1	255	128
1640	DATA	7	255	224
1650	DATA	15	255	240
1660	DATA	31	255	248
1670	DATA	63	255	252
1680	DATA	127	255	254
1690	DATA	255	255	255
1700	DATA	255	255	255
1710	DATA	255	255	255
1720	DATA	255	255	255
1730	DATA	255	255	255
1740	DATA	127	255	254
1750	DATA	63	255	252
1760	DATA	31	255	248
1770	DATA	15	255	240
1780	DATA	7	255	224
1790	DATA	1	255	128
1800	DATA	0	0	0
1810	DATA	0	0	0

Legen Sie sich neben Ihr Raster einen Zettel mit derselben Reiheneinteilung, die das Raster hat. Jede Rasterreihe ist auf der linken Seite numeriert, von Reihe 0 bis Reihe 20. Wie gesagt, erhalten Sie 3 Summen - SUMME AUS A, SUMME AUS B und SUMME AUS C, die jeweils nur aus den Werten der schattierten Pixels in Abschnitt A, B oder C bestehen. Wenn Sie sich diese drei Abschnitte in Reihe 0 anschauen, sehen Sie, daß alle drei leer sind. Ein leerer Abschnitt ist nicht schattiert. Die Summe für einen leeren

Abschnitt beträgt Null (0). Auf der rechten Seite des Rasters sehen Sie drei Bereiche, bezeichnet mit SUMME AUS A, SUMME AUS B und SUMME AUS C:

DATA - Befehle

BASIC-Zeile#	DATA	Summe aus A	Summe aus B	Summe aus C
1610	DATA	0	0	0
	DATA			
	DATA			
	DATA			
	DATA			

In derselben Reihe, in der die Pixels addiert wurden (Reihe 0), notieren Sie eine Null (0) als Summe für jeden Abschnitt.

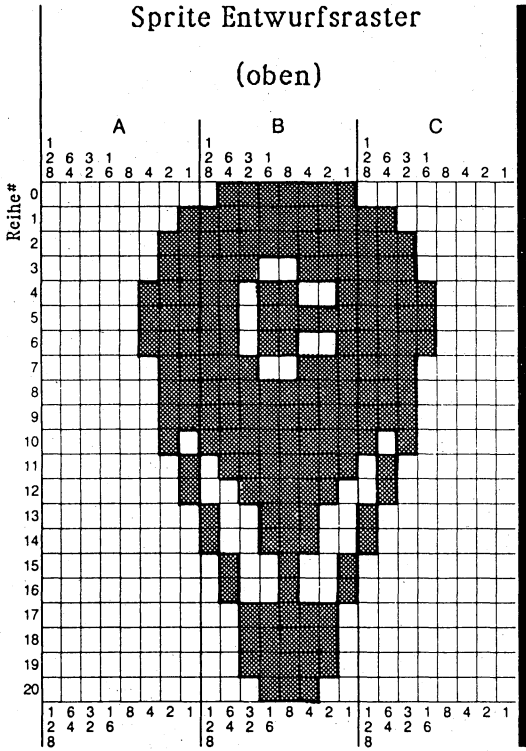
In der nächsten Reihe auf Ihrem Zettel werden die Werte der nächsten Rasterreihe addiert. Sie sehen, daß wieder alle drei Abschnitte leer sind, also notieren Sie 0 als Summe für A, B und C am Ende der Reihe 1.

In der Reihe 2 finden Sie nun einige schattierte Quadrate, die zu addieren sind. Das erste schattierte Quadrat befindet sich in der letzten Spalte des Abschnittes A. Diese Spalte ist mit 1 bezeichnet. Da sich keine weiteren schattierten Pixels in Abschnitt A befinden, ist die SUMME AUS A in dieser Reihe 1. Notieren Sie dies unter SUMME AUS A am Ende der Reihe. In Abschnitt B sind alle Pixels schattiert. Die Summe aus B in dieser Reihe setzt sich aus den Zahlen zusammen, die Sie über jeder Spalte sehen: 128 + 64 + 32 + 16 + 8 + 4 + 2 + 1 = 255. Wenn alle 8 Pixels in einer Reihe innerhalb eines Abschnittes schattiert sind, beträgt die Summe für diese Reihe in diesem Abschnitt immer 255. Notieren Sie unter SUMME AUS B am Ende der Reihe 2 die Zahl 255. In Abschnitt C, Reihe 2 sehen Sie, daß das einzige schattierte Pixel den Wert von 128 hat. Notieren Sie unter SUMME AUS C für diese Reihe die Zahl 128.

Nach dieser einfachen Methode werden alle Summen in jeder Reihe des Rasters ermittelt, wobei, wie bereits erwähnt, ein leerer Bereich immer eine Summe von 0, ein vollständig schattierter Bereich eine Summe von 255 ergibt.

Es ist wichtig, daß jede Summe korrekt errechnet und in die entsprechende Reihe eingetragen wird. Wenn Sie später das Programm eingeben, um das Sprite zu zeichnen, werden diese Summen als DATA-Befehle eingegeben, die dem Computer exakt mitteilen, wie das Sprite aussehen soll.

In Ihrem **Commodore 64** "User's Guide" können Sie sehen, daß die Daten für den Ballon auf dieselbe Weise ermittelt werden. In der folgenden Abbildung ist der Ballon mit seinen Daten abgebildet:



DATA - Befehle

BASIC-Zeile#	DATA	Summe aus A	Summe aus B	Summe aus C
	DATA	0	127	0
	DATA	1	255	192
	DATA	3	255	224
	DATA	3	231	224
	DATA	7	217	240
	DATA	7	223	240
	DATA	7	217	240
	DATA	3	231	224
	DATA	3	255	224
	DATA	3	255	224
	DATA	2	255	160
	DATA	1	127	64
	DATA	1	62	64
	DATA	0	156	128
	DATA	0	156	128
	DATA	0	73	0
	DATA	0	73	0
	DATA	0	62	0
	DATA	0	62	0
	DATA	0	62	0
	DATA	0	28	0

Sehen Sie die nicht schattierten Pixels innerhalb des Ballons? Diese unschattierten Pixels zeigen Teile des Hauptbildes, während der Ballon sich über den Bildschirm bewegt. Diese Pixels sind wie "Fenster", durch die Sie Abbildungen sehen können, die sich darunter befinden. Durch diese Fenster können Sie andere Formen, andere Sprites (falls Sie mehrere in Ihrem Bild haben) und die Hintergrund-

farbe sehen. Dieser Effekt ist eine wirklich einzigartige Eigenschaft von Sprites.

Sie können acht Sprites gleichzeitig in dasselbe Bild einsetzen. Jedes Sprite ist unabhängig von allen anderen Formen in Ihrem Bild, einschließlich anderer Sprites. Jedes Sprite kann ein anderes Aussehen haben, also können Sie auch acht verschiedene Datensätze haben. Sprites können selbstverständlich auch identische Formen haben, so daß in diesem Fall dieselben DATA-Befehle zur Anwendung kommen.

Jedes Sprite ist gekennzeichnet, damit der Computer sich die Sprites in Ihrem Bild merken kann. Diese Kennzeichnung wird als Sprite-Zeiger bezeichnet und besteht aus einer Zahl zwischen 0 und 7. Der Sprite-Zeiger "zeigt" auf eine Stelle im Speicher, an der die Daten des Sprites gespeichert sind. Unser Programm reserviert acht bestimmte Stellen für die Speicherung der Spritedaten.

Wenn ein Sprite definiert ist, kann es eine der sechzehn verfügbaren Spritefarben erhalten. Es kann auch vergrößert, gelöscht und neu positioniert, völlig gelöscht und sogar über den Bildschirm geführt werden. Jedes der acht gleichzeitig möglichen Sprites kann eine andere Form, Größe und Farbe haben.

Spezielle Eigenschaften der Sprites

Sprites haben bestimmte Eigenschaften, über die nur sie verfügen. Eine Liste der Charakteristika beinhaltet:

- Definition der Sprites
- Einschalten/Ausschalten
- X-Erweiterung/X-Reduktion
- Y-Erweiterung/Y-Reduktion
- X- und Y-Erweiterung kombiniert
- Vorrang eines Sprites vor einem anderen Sprite
- Vorrang eines Sprites vor einer Form/einer Form vor einem Sprite
- Spritefarbe
- Plazierung eines Sprites an einer Stelle X,Y auf dem Bildschirm
- Bewegung eines Sprites von X1,Y1 zu X2,Y2

Obwohl die Liste noch einiger Erläuterungen bedarf, bekommen Sie vielleicht schon eine Vorstellung davon, wie groß Ihre Kontrolle über Sprites ist. Die obengenannten Eigenschaften befinden sich in Paketen, genannt (was sonst?) Unteroutinen. Sie werden also 12 weitere Werkzeuge zu Ihrem Werkzeugkasten hinzufügen, bevor Sie an das Ende dieses Kapitels gelangen. Aber keine Angst. Diese Unterroutinen sind sehr kurz, und Sie ersparen Ihnen eine Menge sich wiederholende

Arbeit, die ohne sie erforderlich wäre. In den folgenden Abschnitten werden die Merkmale der Sprites ausführlich erläutert.

Definition der Sprites

Um ein Sprite zu erstellen, müssen Sie zuerst sein Aussehen bestimmen. Dies geschieht durch die Eingabe von DATA-Befehlen, die Reihe für Reihe beschreiben, welche Pixels für die Form des Sprites eingezeichnet werden sollen. Die DATA-Befehle werden gelesen und gespeichert.

Einschalten/Ausschalten

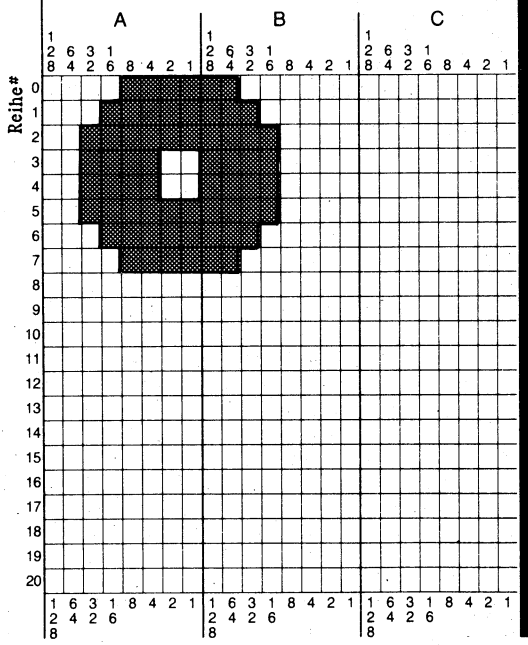
Wenn ein Sprite definiert ist, muß es "eingeschaltet" werden. Die Unterroutine zu dieser Eigenschaft läßt das Sprite sichtbar werden. Beachten Sie, daß ein Sprite auf dem Bildschirm plaziert sein muß (siehe "Plazieren eines Sprites an X,Y"), damit es beim Einschalten sichtbar wird. Drei Schritte sind also notwendig, um ein Sprite zu sehen: Definition, Einschalten und Plazierung auf dem Bildschirm. Um ein Sprite wieder vom Bildschirm verschwinden zu lassen, benutzen Sie das Merkmal "Ausschalten". Ein- und Ausschalten eines Sprites gleicht dem Betätigen eines Lichtschalters. Einige interessante Effekte, wie Blinken, können durch wiederholten Wechsel zwischen diesen Merkmalen erreicht werden.

X-Erweiterung/X-Reduktion

Die Unterroutine zur Eigenschaft "X-Erweiterung" verdoppelt die Breite des Sprites. Dies geschieht durch Verdopplung jeder Spalte des Sprites. Die verdoppelten Spalten erscheinen dann abwechselnd mit den ursprünglichen Spalten, um das breitere Sprite zu bilden. Sehen Sie sich die folgenden beiden Diagramme an, damit Sie dies besser verstehen. Das erste zeigt ein Sprite in seiner ursprünglichen Form, so wie es auf dem Raster entworfen wurde, während das zweite Diagramm zeigt, wie die Spalten auf dem Bildschirm verdoppelt werden. (Die leicht schattierten Spalten sind nur in dieser Weise eingezeichnet, um sie als die Kopien von den anderen zu unterscheiden. Wenn Sie ein Sprite erweitern, werden alle Pixels in derselben Farbe auf dem Bildschirm eingezeichnet.) Sie sehen, daß das ursprüngliche Sprite die ersten zwei Spalten des Rasters nicht belegt. Wenn das Sprite erweitert wird, werden diese beiden Spalten (obwohl sie leer sind) ebenfalls verdoppelt.

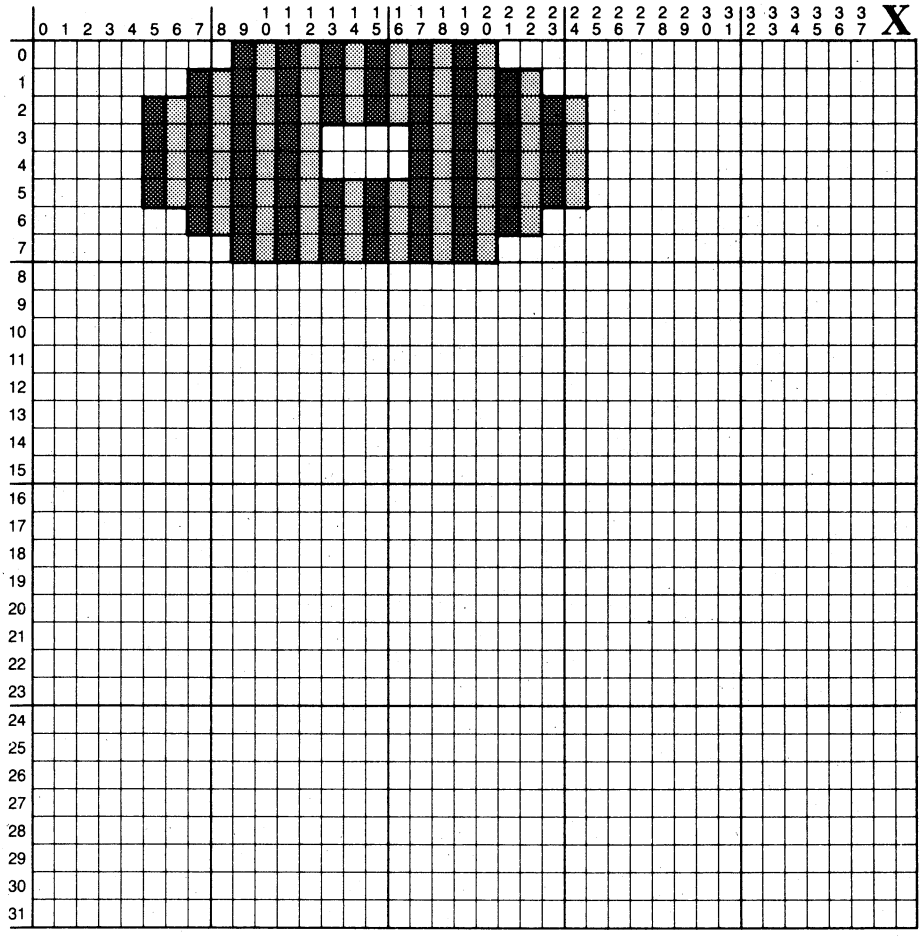
Sprite Entwurfsraster

(oben)



DATA - Befehle

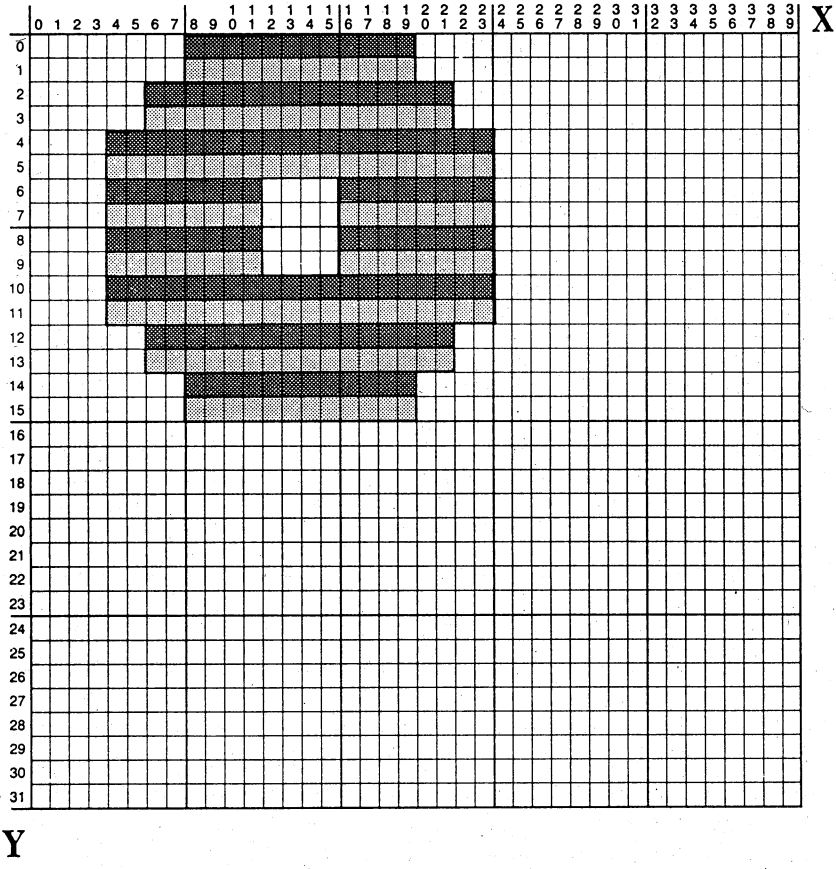
BASIC-Zeile#	DATA	Summe aus A	Summe aus B	Summe aus C



Die Unterroutine zur Eigenschaft "X-Reduktion" entfernt alle kopierten Spalten und gibt dem Sprite seine ursprüngliche Breite zurück. X-Erweiterung dient dazu, Abwechslung in ein Bild zu bringen, das sonst identische Sprites enthält.

Y-Erweiterung/Y-Reduktion

Die Unterroutine zur Eigenschaft "Y-Erweiterung" verdoppelt die Größe eines Sprites in der Höhe. Jede Pixelreihe des Sprites wird verdoppelt. Wenn Sie das x-erweiterte Sprite mit der X-Erweiterung aus dem letzten Diagramm für diesen Vorgang benutzen, erhalten Sie folgende Form:



Die Unterroutine Eigenschaft "Y-Reduktion" entfernt alle kopierten Reihen und gibt dem Sprite seine ursprüngliche Höhe zurück.

X- und Y-Erweiterung kombiniert

Durch Kombination der X- und Y-Erweiterung verdoppelt ein Sprite seine Breite und seine Höhe. Wenn X- und Y-Reduktion kombiniert ausgeführt werden, erhält das Sprite seine ursprüngliche Breite und Höhe zurück.

Vorrang eines Sprites vor einem anderen Sprite

Ein Sprite kann so angelegt werden, daß es vor einem anderen Sprite in einem Bild erscheinen kann., d.h. wenn zwei sich bewegende Sprites aufeinandertreffen, bewegt das eine sich über das andere hinweg weiter. Dieses Sprite, das sich vor das andere schiebt, hat gegenüber dem anderen den größeren Vorrang auf dem Bildschirm als das andere. Der Vorrang bestimmt die Reihenfolge, in der Sprites sichtbar auf dem Bildschirm übereinandergeschichtet werden. Dieser Vorrang wird durch die Nummer der Sprites bestimmt. Das Sprite mit der kleineren Nummer hat immer den Vorrang vor einem mit einer größeren Nummer. Sprite 0 hat beispielsweise den größten Vorrang und erscheint immer vor anderen Sprites auf dem Bildschirm. Sprite 4 hat Vorrang vor den Sprites 5-7. Sprite 7 hat keinen Vorrang vor anderen Sprites.

Sich überschneidende Sprites sind besonders interessant, wenn das vorderste Sprite eine Öffnung oder ein "Fenster" hat. Durch dieses Fenster wird dann das Sprite, das sich hinter dem ersten befindet, sichtbar. Sie sollten sich den Vorrang eines Sprites bei der Entwicklung Ihrer belebten Bilder und vor der Eingabe des Programms überlegen.

Vorrang eines Sprites vor einer Form/einer Form vor einem Sprite

Wenn Sie die Eigenschaft "Vorrang eines Sprites vor einer Form" bei der Sonne in Ihrem Bild verwenden, erscheint sie vor allen anderen Formen des Bildes. Während die Sonne über den Himmel wandert, schiebt sie sich vor das Schiff und die Möven. Diese Art von Vorrang bestimmt, ob ein Sprite vor oder hinter anderen eingezeichneten Formen angezeigt wird, was sich als besonders wirkungsvoll erweist, wenn sie bei einem Sprite mit einem "Fenster" eingesetzt wird. Während das Sprite sich bewegt, sehen Sie andere Formen und die Hintergrundfarbe durch diese Öffnung. Wenn Sie die Eigenschaft "Vorrang einer Form vor einem Sprite" einsetzen, erscheint das Sprite hinter jedem anderen eingezeichneten Objekt auf dem Bildschirm.

Beachten Sie: Ein Sprite hat immer Vorrang vor den Hintergrundfarben des Bildschirms. Wenn es in einen Farbblock gelangt oder einen passiert, erscheint das Sprite an Stelle der Hintergrundpixels, über die es sich bewegt.

Spritefarbe

Die Unterroutine diese Eigenschaft zeichnet ein Sprite in einer einheitlichen Farbe. Sie zeichnet die schattierten Pixels ein, die in Ihrem Entwurfsraster zu Datennummern geändert wurden. Alle Pixels des Sprites werden in derselben Farbe angezeigt.

Sie müssen für jedes einzelne Sprite einen Farbcode bestimmen, auch wenn alle acht Sprites dieselbe Farbe haben. (Wird für ein Sprite keine Farbe im Programm bestimmt, erhält es automatisch die Farbe schwarz.) 16 Spritefarben mit Farbcodes (0 bis 15) stehen Ihnen zur Verfügung:

0 Schwarz	4 Purpur	8 Orange	12 mittleres Grau
1 Weiß	5 Grün	9 Braun	13 Hellgrün
2 Rot	6 Blau	10 Hellrot	14 Hellblau
3 Cyan	7 Gelb	11 Dunkelgrau	15 Hellgrau

Die Farbe jedes Sprites ist unabhängig von jedem anderen Farbenblock im Bild. Sie beeinflußt oder ändert keine Farbe, mit der Pixels, Linien oder Formen eingezeichnet wurden. Ein Sprite kann die Farbe eines Pixels nur vorübergehend ändern, wenn es dieses passiert und Vorrang hat.

Plazieren eines Sprites an X,Y

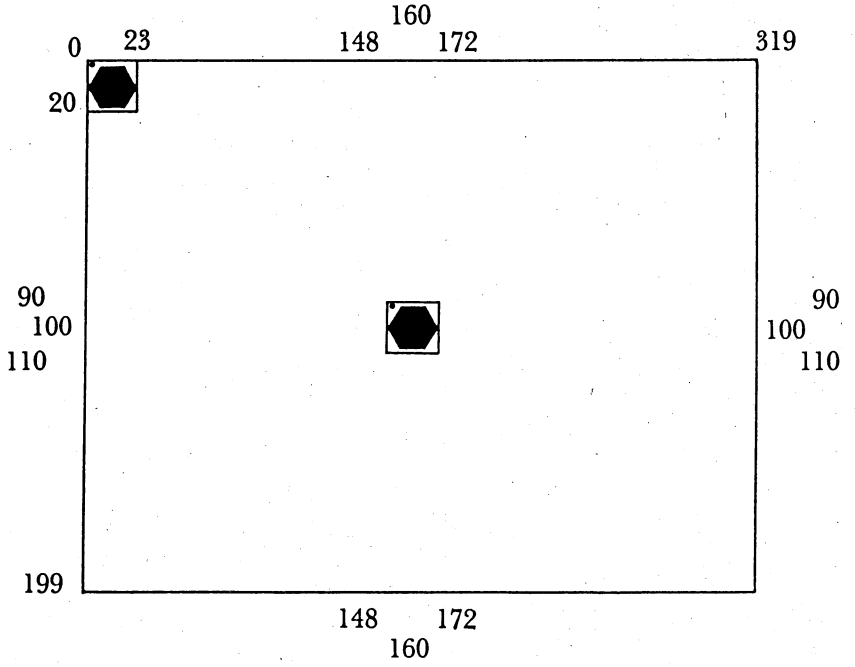
Wenn Sie ein Sprite definieren und "einschalten", müssen Sie es auf dem Bildschirm plazieren. Möglich ist dies an jeder beliebigen Stelle innerhalb und selbst außerhalb des Monitors.

Um ein Sprite zu plazieren, geben Sie für das obere linke Quadrat des Entwurfsrasters die Stelle auf dem Bildschirm an, auch wenn dieses Quadrat nicht von Ihrem Sprite belegt wird. Dieses obere linke Quadrat des Rasters wird als der Nullpunkt des Sprites bezeichnet. Sie erinnern sich, daß Ihre DATA-Befehle alle 504 Quadrate des Rasters beschreiben? Wenn Sie den Nullpunkt korrekt auf dem

Bildschirm plazieren, kann der Computer leicht die restlichen Quadrate in Bezug zu diesem Nullpunkt plazieren.

Um den Nullpunkt zu positionieren, werden ein X- und ein Y-Wert im Programm angegeben. Sie bilden eine X,Y-Koordinate, an deren Stelle der Nullpunkt auf dem Bildschirm plaziert wird. X bestimmt die horizontale und Y die vertikale Plazierung. Stellen Sie sich ein Sprite als ein Blatt Papier vor, das von seinem Nullpunkt-Pixel herumgezogen wird.

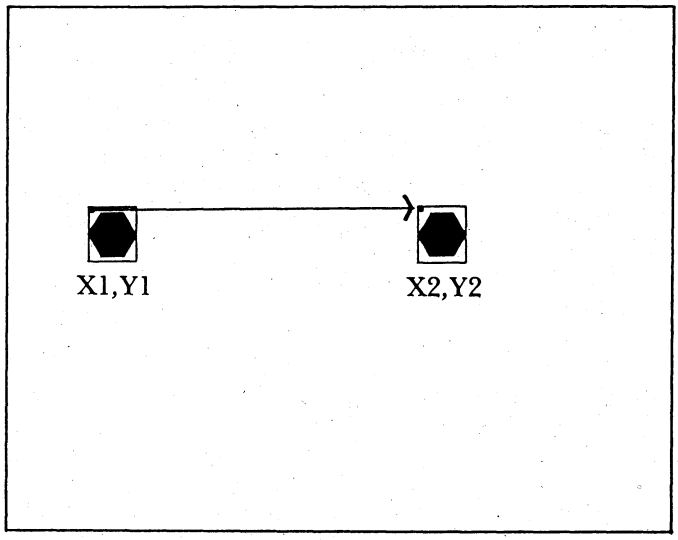
Wird der Nullpunkt neu positioniert, wird das Sprite ebenfalls entsprechend bewegt. Unten sehen Sie als Beispiel zwei Sprites in Form von Sonnen auf dem Bildschirm. Die Kästchen, in denen die Sonnen sich befinden, stellen das "unsichtbare" Raster dar, auf dem das Sprite gezeichnet wurde. Der kleine Punkt in jedem Kästchen stellt den Nullpunkt dar. Der Nullpunkt des Sprites in der oberen linken Ecke wurde bei 0,0 positioniert. Das zweite Sprite wurde in die Mitte des Bildschirms gesetzt, indem für den Nullpunkt die Stelle 148,90 bestimmt wurde.



Stellen Sie sich einfach vor, daß das ganze Entwurfsraster auf dem Bildschirm plaziert wird. Das ganze Raster wird bewegt, wenn Sie eine neue Stelle für den Nullpunkt angeben.

Bewegung eines Sprites von X1,Y1 zu X2,Y2

Sie können ein Sprite vertikal, horizontal und diagonal über den Bildschirm bewegen. Dh., daß das Sprite immer auf dem Bildschirm bleibt, jedoch von einer Stelle zur nächsten bewegt wird. Der Weg, den das Sprite zurücklegt, gleicht einer geraden Linie. Um ein Sprite zu bewegen, geben Sie eine Stelle X1,Y1 und eine Stelle X2,Y2 an. Der Nullpunkt des Sprites wird dann an der Stelle X1,Y1 plaziert und bewegt sich auf geradem Weg zur Stelle X2,Y2. Dabei folgt der Rest des Sprites der Bewegung des Nullpunktes. Beispielsweise könnte die Sonne ihre Bewegung bei Punkt 0,10 (X1 = 0, Y1 = 10) beginnen und zu 319,10 (X2 = 319, Y2 = 10) fortführen. Wie Sie in der folgenden Abbildung sehen, ist es der Nullpunkt, der diesen geraden Weg entlang bewegt wird.



Sprites können in jede Richtung wandern - von links nach rechts, rechts nach links, oben nach unten, unten nach oben und diagonal. Das Sprite bewegt sich auf geradem Weg zwischen dem angegebenen Anfangs- (X1,Y1) und Endpunkt (X2,Y2).

Wenn Sie ein Sprite bewegen, können Sie die Geschwindigkeit seiner Bewegung steuern. Obgleich sich dies schwerlich in Stundenkilometern ausdrücken läßt, können Sprites ganz hübsch aufdrehen. Die Geschwindigkeit wird durch eine Zahl bestimmt, deren Wert "SD"(Sprite-Geschwindigkeit) zugeordnet wird. Dieser Wert teilt dem

Computer mit, wieviele Pixels zwischen jeder Plazierung des Sprites übersprungen werden müssen. Während das Sprite sich über den Bildschirm bewegt, wird es viele Male gelöscht und neu gezeichnet. Wenn Sie 10 für SD einsetzen, löscht der Computer das Sprite und zeichnet es alle 10 Pixels weiter wieder ein. Seien Sie vorsichtig, wenn Sie verschiedene Geschwindigkeiten ausprobieren. Je höher die Geschwindigkeit, desto sprunghafter erscheint die Bewegung des Sprites.

Mit dieser Methode können nicht zwei Sprites gleichzeitig bewegt werden. Dazu müssen Sie zwischen den beiden Sprites wechseln, indem Sie eine Schleife erstellen, die jedes löscht und immer ein Stückchen weiter plaziert.

Im nächsten Abschnitt haben Sie Gelegenheit, diese speziellen Eigenschaften der Sprites auszuprobieren. Wenn Sie einige Erfahrung gesammelt haben, werden Sie bald interessante Trickbilder auf Ihrem **Commodore 64** entwerfen können.

Zeichnen und Plazieren der Sonne

Wenn Sie das Bild aus Kapitel 5 auf den hochauflösenden Bildschirm und das Programm aus Kapitel 5 in den Speicher geladen haben, starten Sie die ZAP-Routine. Geben Sie RUN 10 ein und betätigen Sie RETURN. Wenn Sie keine blinkenden Zeichen mehr auf dem Bildschirm sehen, geben Sie ein:

```
1010 GOSUB 20
6000 GET A$
6010 IF A$ = " " THEN 6030
6020 GOTO 6000
6030 GOSUB 30
6040 END
```

Wie im letzten Kapitel geben Sie das Programm in Abschnitten ein. Nach Eingabe des jeweiligen Abschnitts, überprüfen Sie ihn auf Fehler. Jeder Abschnitt bildet eine Stufe auf dem Weg zur Anfertigung und Bewegung Ihres Sprites, der Sonne. Auf diesem Weg machen wir Ihnen Vorschläge, mit den verschiedenen Eigenschaften der Sprites zu experimentieren. Probieren Sie diese Vorschläge aus, ebenso weitere, die Ihnen zwischendurch einfallen.

Nun geben Sie die folgenden Zeilen der Hauptroutine ein:

```
1100 REM:::::::::SUN SPRITE
1110 SP = 0: GOSUB 120
1120 GOSUB 130
```

```
1130 GOSUB 150
1140 GOSUB 170
1150 GOSUB 200
1160 C = 7: GOSUB 210
1170 X = 232: Y = 10
1180 GOSUB 220
```

Der nächste Abschnitt enthält die DATA-Befehle dieses Sprites, die Sie schon in dem Entwurfsraster gesehen haben. Wenn Sie diesen Abschnitt eingeben, benutzen Sie drei Stellen und ein Komma für jede einzutragende Zahl. Beispielsweise geben Sie die erste Zeile als "DATA l10,l10,l10" ein ("l", nicht zu verwechseln mit 1, bedeutet LEERTASTE, d.h Sie geben vor jeder 0 zwei Leerstellen ein). Mit dieser Methode ordnen Sie die Kommata in gleichmäßigen Spalten an. Diese Anordung der Daten erleichtert Ihnen eine eventuell notwendig werdende Fehlersuche. Geben Sie diesen Abschnitt nun ein:

```
2500 REM:::::::::SUN SPRITE DATA
2510 DATA   0,   0,   0
2520 DATA   0,   0,   0
2530 DATA   1, 255, 128
2540 DATA   7, 255, 224
2550 DATA  15, 255, 240
2560 DATA  31, 255, 248
2570 DATA  63, 255, 252
2580 DATA 127, 255, 254
2590 DATA 255, 255, 255
2600 DATA 255, 255, 255
2610 DATA 255, 255, 255
2620 DATA 255, 255, 255
2630 DATA 255, 255, 255
2640 DATA 127, 255, 254
2650 DATA  63, 255, 252
2660 DATA  31, 255, 248
2670 DATA  15, 255, 240
2680 DATA   7, 255, 224
2690 DATA   1, 255, 128
3000 DATA   0,   0,   0
3010 DATA   0,   0,   0
```

Überprüfen Sie jeden Abschnitt auf Fehler. In jedem der 21 DATA-Befehle müssen drei Zahlen stehen. Wenn Sie eine Nummer ausgelassen haben, muß die betreffende Zeile kürzer als die andere sein. Haben Sie eine Zahl hinzugefügt, muß die betreffende Zeile

länger als die anderen sein. Korrigieren Sie alle Fehler, dann geben Sie die erste neue Unterroutine ein:

```
120 REM:::::::::DEFINE SPRITE SP
121 FOR I = 0 TO 62
122 READ A
123 POKE 16384 + 64*SP + I,A
124 NEXT I
125 POKE 18424 + SP,SP
126 RETURN
```

Sehen Sie sich jede Zeile an. Korrigieren Sie alle Fehler. Gönnen Sie Ihren Augen eine kleine Pause, dann fahren Sie mit der Eingabe der folgenden Zeilen fort:

```
130 REM:::::::::TURN ON SPRITE SP
131 POKE 53269, PEEK(53269) OR 2^SP
132 RETURN
140 REM:::::::::TURN OFF SPRITE SP
141 POKE 53269, PEEK(53269) AND (255-2^SP)
142 RETURN
150 REM:::::::::X EXPAND SPRITE SP
151 POKE 53277, PEEK(53277) OR 2^SP
152 RETURN
160 REM:::::::::X UNEXPAND SPRITE SP
161 POKE 53277, PEEK(53277) AND (255-2^SP)
162 RETURN
```

Machen Sie eine weitere kleine Pause und überprüfen Sie Ihre Eingabe. Wenn alles in Ordnung ist, fahren Sie fort:

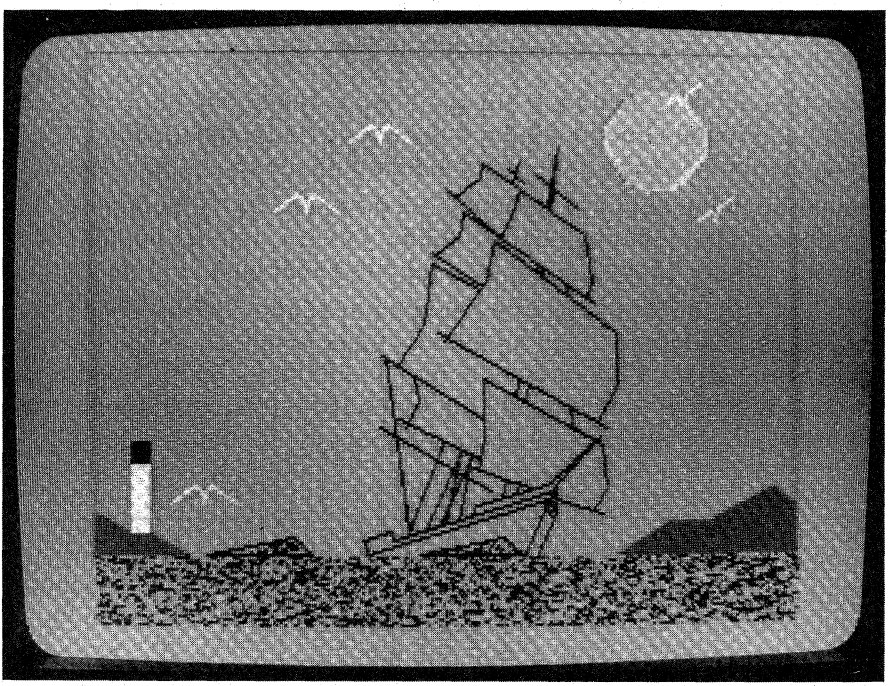
```
170 REM:::::::::Y EXPAND SPRITE SP
171 POKE 53271, PEEK(53271) OR 2^SP
172 RETURN
180 REM:::::::::Y UNEXPAND SPRITE SP
181 POKE 53271, PEEK(53271) AND (255-2^SP)
182 RETURN
190 REM:::::::::SPRITE SP PRIORITY OVER SHAPE
191 POKE 53275, PEEK(53275) AND (255-2^SP)
192 RETURN
200 REM:::::::::SHAPE PRIORITY OVER SPRITE SP
201 POKE 53275, PEEK(53275) OR 2^SP
202 RETURN
210 REM:::::::::SET SPRITE SP TO COLOR C
211 POKE 53287 + SP,C
212 RETURN
```

Auf zum Endspurt. Wenn Sie die Zeilen bis 212 geprüft haben, geben Sie diese ein:

```
220 REM::::::::::PLACE SPRITE SP AT X,Y
221 XX = X + 24: YY = Y + 50:Z% = XX/256
222 V = XX - Z%*256: W = 53248 + SP*2
223 WW = 53264
224 PR = ABS((PEEK(WW) AND 2^SP) <>0)
225 VV = PEEK(WW) AND (255-2^SP) OR (2^SP*Z%)
226 IF PR<>Z% THEN GOSUB 140
227 POKE W,V: POKE WW,VV: GOSUB 130
228 POKE 53249 + SP*2, YY
229 RETURN
```

Sehen Sie Ihre Eingabe noch einmal durch, bevor Sie das Programm starten. Wenn so viel eingegeben wird, können leicht Fehler auftreten. Starten Sie nun das Programm.

Wenn das Programm fehlerfrei läuft, sehen Sie Ihr Sprite in Form einer gelben Sonne oben rechts auf dem Bildschirm erscheinen:



Schauen Sie sich Ihr Sprite an. Befindet es sich an derselben Stelle wie in unserer Abbildung? Es muß einfarbig gelb sein. Wenn Sie Schwierigkeiten haben, gehen Sie folgendermaßen vor:

Wenn das Sprite überhaupt nicht erscheint, überprüfen Sie:

- die Zeilen 1110 und 1120, die das Sprite definieren und einschalten,
- die Zeilen 1170 und 1180, die das Sprite auf dem Bildschirm plazieren,
- die Unterroutinen in den Zeilen 120 bis 132.

Wenn das Sprite deformiert ist, überprüfen Sie:

- Ihre DATA-Befehle in den Zeilen 2510 bis 3010 und die Schleife in Zeile 121,
- die Zeilen 1130 und 1140, die das Sprite in Breite und Höhe erweitern,
- die Unterroutinen in den Zeilen 150-152 und 170-172.

Wenn das Sprite die falsche Farbe hat, überprüfen Sie:

- Zeile 1160, die den Farbcode bestimmt und eine Unterroutine aufruft,
- die Unterroutine in den Zeilen 210-212.

Sie brauchen das Bild aus Kapitel 5 nicht neu laden, bevor Sie ein korrigiertes Programm starten. Solange in Zeile 1110 SP=0 ist und die Zeilen 1170 und 1180 die Position des Sprites auf dem Bildschirm angeben, wird Sprite 0 nach den Angaben des korrigierten Programms neu positioniert.

Wenn Sie sehen, daß das Programm korrekt läuft, betätigen Sie die LEERTASTE und listen Sie die Zeilen 1100 und 1110.

```
1100 REM:::::::::SUN SPRITE
1110 SP = 0: GOSUB 120
```

Zeile 1110 hat drei Aufgaben. Als erstes bestimmt sie, daß alle Programmzeilen, die Sprites betreffen, nur Sprite 0 beeinflussen, bis der Wert von SP geändert wird. Die Sprite-Unterroutinen beeinflussen nur das Sprite, dessen Nummer dem aktuellen Wert von SP entspricht.

Als zweites erhält das Sprite (die Sonne) mit der Nummer 0 den Vorrang vor allen Sprites, die möglicherweise später noch in dem Bild plaziert werden. Wie schon gesagt, können Sie bis zu acht verschiedene Sprites in ein Bild setzen, vorausgesetzt, jedes erhält eine eigene Nummer (von 0 bis 7).

Als drittes definiert GOSUB 120 das Sprite 0 im Speicher. Dazu liest das Werkzeug 120 eine Sammlung von DATA-Befehlen. Es beginnt bei der ersten ungelesenen Dateneinheit im Programm. Die

Schleife READ wird für 63 Dateneinheiten bestimmt, deshalb müssen Sie sich vergewissern, daß Sie alle 63 Einheiten in Ihr Programm eingegeben haben. Listen Sie die Zeilen 120-126. Im folgenden abgetrennten Abschnitt ist die Unterroutine, die ein Sprite definiert, erklärt.

Werkzeug 120::::::::::Sprite-Umrisse

```
120 REM:::::::::DEFINE SPRITE SP
121 FOR I = 0 TO 62
122 READ A
123 POKE 16384 + 64*SP + I,A
124 NEXT I
125 POKE 18424 + SP,SP
126 RETURN
```

Zweck: *Dieses Werkzeug definiert die Form für das Sprite. Es liest die Sprite-Daten und speichert die Form des Sprite im Computerspeicher. Es plaziert das Sprite nicht auf dem Bildschirm (siehe Werkzeug 220 für Plazierung der Sprites).*

Anwendung: *Um ein Sprite zu definieren, müssen Sie folgendermaßen vorgehen:*

1. *Zeichnen Sie das Sprite auf dem "Sprite-Entwurfsraster".*
2. *Addieren Sie drei Summen (A,B,C) für die schattierten Pixels in jeder Reihe des Rasters.*
3. *Es müssen sich drei Zahlen (Datensummen) in jeder der 21 Reihen ergeben. Geben Sie diese Zahlen als DATA-Befehle in der Hauptroutine ein.*
4. *Geben Sie in der Hauptroutine einen Befehl ein, um das Sprite mit einer Nummer (0-7) zu identifizieren. Hinter dieser Nummer geben Sie GOSUB 120 ein, z.B.: 1110 SP = 0: GOSUB 120*

Technische Beschreibung: *Sprite-Daten werden am Anfang von Bank 1 gespeichert. Jedes Sprite benötigt 64 Bytes an Speicherplatz für die Speicherung der Bilddefinition. Für 8 Sprites werden also 512 freie Bytes des Speicherplatzes benötigt (64x8 = 512). Da Bank 1 am Anfang 512 Bytes hat, eignet sich diese Stelle gut zur Speicherung der Sprite-Daten.*

```
121 FOR I = 0 TO 62
122 READ A
123 POKE 16384 + 64*SP + I,A
124 NEXT I
```

*Diese Zeilen lesen zuerst die Sprite-Daten aus Ihren DATA-Befehlen. Dann speichern Sie die Daten in den entsprechenden Speicherstellen für das angegebene Sprite. Zeile 123 berechnet die richtige Speicherstelle für jedes Sprite. "16384" ist die erste Speicherstelle in Bank 1. "64*SP" bewirkt, daß die Daten in den richtigen 64 Bytes für das Sprite (SP) gespeichert werden. "I" wird für jede Schleife erhöht, um die Sprite-Daten in aufeinanderfolgende Speicherstellen zu plazieren.*

```
125 POKE 18424 + SP,SP
```

Zeile 125 bestimmt einen Zeiger, der auf das Sprite zeigt, das gerade eingelesen wurde. Jeder Satz von Sprite-Daten wird in einem Block gespeichert. Die ersten 64 Bytes in Bank 1 bilden Block 0. Die zweiten 64 Bytes bilden Block 1.

Jedes Sprite hat seinen eigenen Block:

Sprite 0 --> Block 0
Sprite 1 --> Block 1
Sprite 2 --> Block 2 (etc.)

Zeile 125 ordnet jedem Sprite seinen eigenen Block zu.

Listen Sie Zeile 1120. Diese Zeile benutzt das Werkzeug 130, um das Sprite im Speicher "einzuschalten". Auf dem Bildschirm sichtbar wird es erst, wenn es vermittels einer X,Y-Koordinate plaziert worden ist. Sobald das geschehen ist, funktionieren die Werkzeuge "Einschalten" und "Ausschalten" wie Schalter zum Auftauchen/Löschen des Sprites auf dem Bildschirm. Wenn ein Sprite "ausgeschaltet" ist, verschwindet es vom Bildschirm. Sie können bis zu 8 Sprites in ein Bild einsetzen. Dabei können einige "ein-" oder "ausgeschaltet" sein, je nachdem, wie Sie Ihr Bild entworfen haben. Listen Sie die Zeilen 130-132.

Werkzeug 130::::::::::Sprite-Einschalten

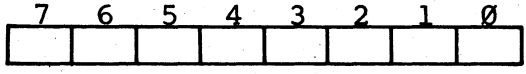
```
130 REM:::::::::TURN ON SPRITE SP
131 POKE 53269, PEEK(53269) OR 2^SP
132 RETURN
```

Zweck: *Dieses Werkzeug "schaltet" ein Sprite im Speicher "ein".*

Anwendung: *Nachdem ein Sprite mit Hilfe des Werkzeugs 120 definiert ist, schaltet das Werkzeug 130 es ein. Wenn das Sprite auf dem Bildschirm plaziert worden ist, wird es durch den Einsatz dieses Werkzeugs sichtbar. Um das Werkzeug zu benutzen, setzen Sie SP mit der Nummer des einzuschaltenden Sprites gleich und geben dann den Befehl GOSUB 130 ein.*

Technische Beschreibung: *Vorweg die Definition eines neuen Begriffes: Register. Die meisten Speicherstellen sind einfach Lagerplätze für Zahlen. Ein Register ist eine spezielle Speicherstelle mit einer bestimmten Funktion. Die Zahl, die in einem Register gespeichert ist, kann sehr spannende Resultate bewirken. Beispielsweise schalten die Unterroutinen in den Zeilen 20 und 30 die hochauflösende Grafik ein und aus. Die Speicherstellen, die von solchen Unterroutinen benutzt werden, sind Beispiele für Register.*

Die Unterroutinen in den Zeilen 120 bis 202 ändern einige Registerzahlen, um Sprites zu beeinflussen. Einige Register wirken nur auf ein Sprite ein, andere auf alle acht Sprites. Die Farbenunterroutine in Zeile 210 bestimmt die Farben für jedes Sprite. Dies ist ein Beispiel für eine Unterroutine, bei der jedes Sprite sein eigenes Register hat. Die Unterroutinen in den Zeilen 120 bis 202 sind Beispiele für Register, die auf alle acht Sprites einwirken. Sehen wir uns zum besseren Verständnis das Schema eines Registers an. Jedes Register hat acht "Bits", die von rechts nach links mit 0-7 numeriert sind.



Jedem der acht möglichen Sprites wird ein Bit zugeordnet, welches das Sprite steuert. Sprite 0 erhält Bit 0, Sprite 1 erhält Bit 1 usw. Jedes Bit enthält entweder eine 0 oder eine 1. Diese Bestimmung funktioniert hervorragend bei Eigenschaften, die nur einen Zustand von zwei möglichen zulassen, wie ein oder aus, erweitert

oder nicht erweitert. Jeder Zustand wird durch den Wert des Bits bestimmt.

Register 53269 bestimmt, mit welchem Sprite der Computer gerade arbeiten soll. Ein auf 1 geschaltetes Bit bedeutet, daß die Eigenschaften und die Plazierung des entsprechenden Sprites wirksam sind. Wenn alle Bits des Registers 53269 auf 0 geschaltet sind, bedeutet dies, daß der Computer die zugehörigen Sprites nicht anzeigt. Um die Bits zu ändern, benötigen wir einen Befehl, der dem Computer etwa folgendes mitteilt:

ÄNDERE BIT 5 IN REGISTER 53269 ZU 0 oder
ÄNDERE BIT 5 IN REGISTER 53269 ZU 1

Leider gibt es solch einen Befehl nicht. Wir können diese Mitteilungen aber mit BASIC-Befehlen eingeben.

POKE REGISTER #, PEEK (REGISTER #) OR 2^BIT

(BIT = 0 BIS 7)

Mit obiger Befehlsform können wir das angegebene Bit auf 1 schalten, während alle anderen Bits in ihrem jeweiligen Zustand verbleiben.

POKE REGISTER #, PEEK (REGISTER #) AND (255 - 2^BIT)

Mit dieser Befehlsform können wir das angegebene Bit auf 0 schalten, während alle anderen Bits unangetastet bleiben.

Die Unterroutinen in den Zeilen 130 bis 202 enthalten je einen BASIC-Befehl in einer der oben angegebenen Form, je nach dem gewünschten Zustand des Registers. Beachten Sie, daß in den Programmzeilen SP (Abkürzung für Sprite-#) anstelle von "BIT" verwendet wird, da beides dasselbe ist (Sprite 0 = Bit 0, Sprite 1 = Bit 1, Sprite 2 = Bit 2 etc.).

Wie die Funktion "Einschalten" eines Sprites, existiert ebenfalls die gegenteilige Funktion: "Ausschalten", die das Sprite vom Bildschirm verschwinden läßt. Listen Sie die Zeilen 140-142. Werkzeug 140 löscht ein Sprite genauso einfach, wie 130 es umgekehrt einzuschalten vermag. Sie müssen nur angeben, welches Sprite gelöscht werden soll (SP = ?) und den Befehl GOSUB 140 anfügen.

Werkzeug 140::::::::::Sprite-Ausschalten

```
140 REM:::::::::TURN OFF SPRITE SP
141 POKE 53269, PEEK(53269) AND (255-2^SP)
142 RETURN
```

***Zweck**: Das Werkzeug entfernt ein Sprite ("schaltet es aus") vom Bildschirm.*

***Anwendung**: Um dieses Werkzeug zu benutzen, müssen Sie das auszuschaltende Sprite angeben, indem Sie SP mit der entsprechenden Sprite-Nummer (0-7) gleichsetzen, worauf der Befehl GOSUB 140 folgen muß.*

***Technische Beschreibung**: Diese Unterroutine benutzt die zweite Befehlszeile, die wir in der Erläuterung der Unterroutine TURN ON SPRITE vorgestellt haben. Wenn Sie diese Erläuterungen noch nicht verstanden haben, lesen Sie sie noch einmal durch, bevor Sie fortfahren.*

Die Unterroutine 140 setzt ein Bit auf 0. Wird dieser Vorgang im entsprechenden Register und für das richtige Bit ausgeführt, wird das Sprite ausgeschaltet. Diese Unterroutine ist das Gegenteil zur vorherigen, die das Bit auf 1 setzte, um das Sprite einzuschalten. Der Befehl der das Bit ausschaltet wird in folgender Form eingegeben:

POKE REGISTER #, PEEK (REGISTER #) AND (255 - 2^SP)

Denken Sie daran, daß SP die auszuschaltende Bit/Sprite-Nummer ist (0-7).

Beachten Sie, daß das Register in dieser und der vorherigen Unterroutine dasselbe ist: 53269. Dieses Register hat die spezielle Aufgabe, Sprites "ein-" und "auszuschalten".

Da das Werkzeug 130 ein Sprite einschaltet und das Werkzeug 140 ein Sprite ausschaltet, können Sie ein blinkendes Sprite erstellen, indem Sie diese Werkzeuge abwechselnd für dasselbe Sprite einsetzen. Das Sprite erscheint, verschwindet, erscheint und verschwindet wieder in schneller Folge. Diese Technik könnte bei Bildern mit Feuer oder einem Feuerschweif hinter einer Rakete eingesetzt werden, ebenso für eine blinkende Ampel oder flimmernden Sternen bei Nacht.

Listen Sie die Programmzeilen 1130 und 150-152. In Zeile 1130 setzt der Befehl GOSUB 150 das Werkzeug 150 ein, um die Breite

des Sprites zu erweitern. Dieses Werkzeug verdoppelt die Größe der Sonne in der Richtung X.

Werkzeug 150::::::::::X Sprite-Erweiterung

```
150 REM::::::::::X EXPAND SPRITE SP
151 POKE 53277, PEEK(53277) OR 2^SP
152 RETURN
```

Zweck: *Dieses Werkzeug erweitert die Breite eines Sprites auf das Doppelte.*

Anwendung: *Um dieses Werkzeug zu benutzen, müssen Sie SP mit der Nummer des zu vergrößernden Sprites gleichsetzen (SP = ?), worauf der Befehl GOSUB 150 folgen muß.*

Technische Beschreibung: *Die Unterroutine benutzt Register 53277. Dieses Register steuert die Ausweitung und Schrumpfung der Breite eines Sprites. Jedes Sprite kann entweder normale Größe haben oder in der Richtung X erweitert werden. Wenn ein Bit im betreffenden Register gleich 0 ist, hat das entsprechende Sprite normale Breite. Ist das Bit gleich 1, wird die Breite des Sprites erweitert.*

Dieses Werkzeug benutzt die erste der bei Werkzeug 130 vorgestellten Befehlsformen, um Bits auf 1 zu schalten. Dies erweitert sofort das angegebene Sprite in der Richtung X. Ein Sprite muß nicht eingeschaltet sein, um es erweitern zu können.

Die Größe jedes Sprites wird einzeln gesteuert. Um ein Sprite zu erweitern, müssen Sie das richtige Sprite angeben und dann die Unterroutine 150 aufrufen. Um das Sprite wieder zu seiner ursprünglichen Größe zu ändern, müssen Sie es "reduzieren". Dazu bestimmen Sie für SP die richtige Spritenummer und geben dann den Befehl GOSUB 160 in das Programm ein. Probieren Sie die Eigenschaft "X-Reduktion" aus, indem Sie die Zeile 1130 folgendermaßen ändern:

```
1130 GOSUB 160 RETURN
```

Starten Sie das Programm mit dieser Änderung. Das Sprite erscheint auf Grund der "Reduktion" in der halben Breite ziemlich schmal.

Werkzeug 160::::::::::X Sprite-Reduktion

```
160 REM::::::::::X UNEXPAND SPRITE SP
161 POKE 53277, PEEK(53277) AND (255-2^SP)
162 RETURN
```

Zweck: *Dieses Werkzeug verkleinert ein erweitertes oder vergrößertes Sprite zurück auf seine ursprünglichen Breite. Das Werkzeug verändert hingegen kein Sprite, welches nicht vorher mit dem Werkzeug 150 erweitert worden ist.*

Anwendung: *Um dieses Werkzeug zu benutzen, müssen Sie SP mit der Nummer des zu reduzierenden Sprites gleichsetzen. Dieser Angabe muß der Befehl GOSUB 160 folgen.*

Technische Beschreibung: *Diese Unterroutine ist das Gegenteil zu X EXPAND SPRITE (Werkzeug 150). Sie benutzt die zweite, bei Unterroutine 130 vorgestellte Befehlszeile. Das durch den aktuellen Wert von SP angegebene Sprite erhält seine normale Breite zurück, indem das entsprechende Bit in Register 53277 auf 0 gesetzt wird.*

Betätigen Sie die LEERTASTE, um wieder Textanzeige zu erhalten. Listen Sie die Zeilen 1140 und 170-172 auf Ihrem Bildschirm. Der Befehl GOSUB 170 erweitert Ihr Sprite vertikal (nach oben und unten). Die Sonne hat aufgrund der Verwendung des Werkzeugs 170 die doppelte Höhe (im Vergleich zu Ihrem Entwurf). Die Unterroutine Y EXPAND SPRITE verdoppelt jede Pixelreihe des Sprites, so daß das Sprite die doppelte Höhe erhält.

Werkzeug 170::::::::::Y Sprite-Erweiterung

```
170 REM::::::::::Y EXPAND SPRITE SP
171 POKE 53271, PEEK(53271) OR 2^SP
172 RETURN
```

Zweck: *Bei der Benutzung dieses Werkzeugs verdoppelt ein Sprite seine Höhe. Das geschieht durch Verdopplung jeder Pixelreihe des Sprites, beginnend bei der obersten Reihe. Wenn das Werkzeug bei einem Sprite eingesetzt wird, das schon in der Höhe erweitert ist, geschieht nichts.*

Anwendung: *Um dieses Werkzeug zu benutzen, müssen Sie SP mit der Nummer des zu erweiternden Sprites gleichsetzen. Dieser Angabe muß der Befehl GOSUB 170 folgen.*

Technische Beschreibung: *Diese Unterroutine ist identisch mit der Unterroutine X EXPAND SPRITE, mit Ausnahme des Registers 53271, das die Höhe und nicht die Breite des Sprites bestimmt. Dieses Werkzeug schaltet das entsprechende Bit ("SP") ein, so daß der Computer die Höhe des Sprites durch Kopieren jeder Pixelreihe verdoppelt.*

Sprites können die "normale" Höhe, die durch die ursprüngliche angegebenen Daten definiert ist, oder eine erweiterte Höhe durch die Benutzung des Werkzeugs 170 bekommen. Sie können beide Eigenschaften, Erweiterung der Breite und der Höhe, nur eine oder überhaupt keine der beiden Eigenschaften einsetzen.

Um das Sprite wieder in seiner ursprünglichen Höhe erscheinen zu lassen, wird das Werkzeug 180 eingesetzt. Ändern Sie die Zeile 1140 folgendermaßen, um die Höhe der Sonne zu reduzieren:

```
1140 GOSUB 180 RETURN
```

Starten Sie das Programm.

Die Sonne erscheint nun in ihrer ursprünglichen Höhe, wie sie in den DATA-Befehlen definiert ist. Weitere Informationen zu dieser Eigenschaft der Sprites finden Sie in der folgenden technischen Beschreibung.

Werkzeug 180::::::::::Y Sprite-Reduktion

```
180 REM::::::::::Y UNEXPAND SPRITE SP
181 POKE 53271, PEEK(53271) AND (255-2^SP)
182 RETURN
```

Zweck: *Durch die Benutzung dieses Werkzeugs wird ein in der Höhe erweitertes Sprite wieder auf seine ursprüngliche Höhe gebracht, wie sie in den DATA-Befehlen definiert ist. Wenn dieses Werkzeug für ein Sprite benutzt wird, das schon seine normale Höhe hat, geschieht nichts.*

Anwendung: *Um dieses Werkzeug zu benutzen, müssen Sie SP mit der Nummer des zu verkleinernden Sprites gleichsetzen (0-7) und dann den Befehl GOSUB 180 in die Hauptroutine eingeben.*

Technische Beschreibung: *Unterroutine 180 ist identisch mit der Unterroutine X UNEXPAND SPRITE, mit Ausnahme des Registers 53271, das die Höhe und nicht die Breite des Sprites bestimmt.*

Dieses Werkzeug schaltet das entsprechende Bit ("SP") ein, so daß der Computer das Sprite in seiner ursprünglichen Höhe anzeigt.

Kehren Sie zurück zur Textanzeige und listen Sie die Zeilen 1150 und 200-202. Zeile 1150 ruft die Unterroutine in Zeile 200 auf. Dieses Werkzeug gibt den Vordergrundpixels einer Form Vorrang vor dem durch den aktuellen Wert von SP angegebenem Sprite. Ein Sprite hat aber immer Vorrang vor Hintergrundpixels. Dies bedeutet für Ihr Bild, daß eine Form vor der Sonne erscheint, wenn diese an derselben Stelle plaziert wird, an der sich die Form befindet. In Ihrem Bild erscheint die Seemöve vor der Sonne.

Plazieren wir die Sonne nun an einer Stelle, an der Wasser eingezeichnet ist, um zu sehen, was hier den Vorrang hat. Setzen Sie die Sonne an die Stelle X = 232, Y = 166, indem Sie die folgende Zeile eingeben:

```
1170 X = 232: Y = 166
```

Starten Sie das Programm mit dieser Änderung. Die Sonne erscheint nun hinter dem rechten Landstück und im Wasser aufgrund der Tatsache, daß nur die Vordergrundpixels Vorrang vor dem Sprite haben. Das Wasser besteht zu 70% aus Hintergrundpixels, vor denen das Sprite Vorrang hat. Kehren Sie zur Textanzeige zurück und bewegen Sie das Sprite noch einmal:

```
1170 X = 207: Y = 166
```

Wenn Sie nun das Programm starten, sieht es so aus, als ob die Sonne über dem Wasser untergeht. Das Wasser scheint das Bild der Sonne zu reflektieren, wie bei einem Sonnenuntergang. Ein schöner Effekt, den Sie ebensogut bei anderen Bildern einsetzen können!

Sie können Formen Vorrang vor einem, mehreren oder keinen Sprites geben. Um Formen Vorrang vor einem Sprite zu geben, müssen Sie das Sprite angeben (SP = ?) und dann diese Unterroutine mit dem Befehl GOSUB 200 aufrufen.

Werkzeug 200::::::::::Vorrang einer Form vor einem Sprite

```
200 REM:::::::::SHAPE PRIORITY OVER SPRITE SP
201 POKE 53275, PEEK(53275) OR 2^SP
202 RETURN
```

Zweck: *Dieses Werkzeug gibt den Formen Vorrang vor einem angegebenen Sprite (SP = ?). Wenn ein Sprite und eine Form an derselben Stelle plaziert werden, erscheint das Sprite nur an der Stelle, an der sich keine Vordergrundpixels der Form befinden.*

Anwendung: *Um dieses Werkzeug zu benutzen, müssen Sie SP mit der Nummer des betreffenden Sprites gleichsetzen und den Befehl GOSUB 200 in Ihrer Hauptroutine folgen lassen.*

Technische Beschreibung: *Register 53275 bestimmt den Vorrang des Sprites/der Form. Ist ein Bit auf 1 geschaltet, erscheint das entsprechende Sprite hinter den auf dem Bildschirm eingezeichneten Formen (z.Bsp. dem Schiff oder Land). Ist das Bit auf 0 geschaltet, erscheint das entsprechende Sprite vor diesen Formen. (Wie gesagt, bedeutet "Form" hier immer Vordergrundpixels).*

Um ein Sprite hinter einer Form erscheinen zu lassen, müssen Sie die erste der Befehlszeilen benutzen, die wir im Zusammenhang mit der Unterroutine TURN ON SPRITE erklärt haben.

Betätigen Sie die LEERTASTE. Geben wir nun der Sonne den Vorrang vor den Formen und plazieren sie wieder über dem Land. Dazu geben Sie die folgenden Zeilen ein:

```
1150 GOSUB 190
1170 X = 232: Y = 166
```

Starten Sie das Programm mit dieser Änderung. Die Sonne erscheint vor dem Wasser und dem rechten Landstück, da sie Vorrang vor den Formen hat, wenn die Unterroutine 190 aufgerufen wird. Hätte das Sprite eine Öffnung, könnten Sie die Formen hinter der Sonne durch dieses "Fenster" sehen. Setzen wir solch ein Fenster einmal in die Sonne - über Änderungen in den DATA-Befehlen - ein. Betätigen Sie die LEERTASTE, um Textanzeige zu erhalten.

Um eine begrenzte horizontale Öffnung in die Sonne zu setzen, ändern Sie einige Dateneinheiten in den Zeilen 2590 bis 2630. Diese Zeilen definieren den mittleren Bereich des Sprites. Listen Sie diese Zeilen auf dem Bildschirm. Im Moment lauten alle DATA-Befehle in diesen Zeilen 255, weil alle entsprechenden Pixels in Ihrem Entwurfsraster schattiert sind. Ändern Sie diese Zeilen folgendermaßen:

```
2590 DATA 255,   0, 255
2600 DATA 255,   0, 255
2610 DATA 255,   0, 255
```

```
2620 DATA 255,   0, 255
2630 DATA 255,   0, 255
```

In jeder Zeile stellt die Null eine leere Stelle innerhalb des Sprites dar. Diese nicht benutzten Pixels können Sie sich als "transparent" vorstellen, da an diesen Stellen die Formen hinter dem Sprite erscheinen. Starten Sie das Programm.

Sie sehen das Wasser und das Land durch dieses Fenster. Das Werkzeug 190 kann einige interessante Effekte mit Sprites erzeugen - speziell wenn das Sprite sich bewegt. (Sie werden Ihre Sonne in Kürze bewegen.)

Werkzeug 190::::::::: Vorrang eines Sprite vor einer Form

```
190 REM:::::::::SPRITE SP PRIORITY OVER SHAPE
191 POKE 53275, PEEK(53275) AND (255-2^SP)
192 RETURN
```

Zweck: *Wenn ein Sprite Vorrang vor einer Form hat, erscheint es vollständig vor einer Form, die sich an derselben Stelle wie das Sprite befindet. Die Benutzung dieses Werkzeugs bringt insbesondere bei Sprites mit Fenstern Spaß, da Sie die Formen durch die Öffnung sehen können.*

Anwendung: *Um dieses Werkzeug zu benutzen, müssen Sie SP mit der Nummer des entsprechenden Sprites gleichsetzen und den Befehl GOSUB 190 in der Hauptroutine folgen lassen.*

Technische Beschreibung: *Diese Unterroutine benutzt die zweite Befehlszeile, die anläßlich der Unterroutine TURN ON SPRITE beschrieben wurde. Register 53275 bestimmt das Sprite, welches Vorrang vor den Formen hat, indem dessen Bit auf 0 geschaltet wird.*

Um fortzufahren, betätigen Sie die LEERTASTE. Schauen wir uns nun die Unterroutine für die Farbe des Sprites an. Listen Sie die Zeilen 1160 und 210-212. In Zeile 1160 wird der Sprite-Farbcode 7, gelb, mit der Farbvariablen "C" gleichgesetzt. Mit dem zweiten Befehl, GOSUB 210, wird das Sprite gezeichnet. Mit Hilfe dieses Werkzeugs werden alle entsprechenden Pixels innerhalb des Sprites zur mit C angegebenen Farbe geändert. Sie können die Pixels in einem Sprite nicht in verschiedenen Farben zeichnen. Sechzehn Spritefarben (0-15) stehen Ihnen zur Verfügung. (Sie finden die Farben mit den Codenummern weiter vorne in diesem Kapitel.) Um die Farbe eines

Sprites zu ändern, müssen Sie für C einen anderen Code angeben. Probieren Sie die Farbe Rot (2) für Ihre Sonne aus, indem Sie eingeben:

```
1160 C = 2: GOSUB 210
```

Starten Sie das Programm wieder. Experimentieren Sie mit weiteren Farben, indem Sie den Farbcode in Zeile 1160 ändern und das Programm wieder starten. Spritefarben benutzen denselben Farbcode (C), wie Ihre Formen, deshalb müssen Sie den Wert von C jedesmal neu bestimmen, wenn Sie ein neues Sprite oder eine neue Form zeichnen wollen.

Werkzeug 210::::::::::Farbgebung für ein Sprite

```
210 REM:::::::::SET SPRITE SP TO COLOR C
211 POKE 53287 + SP,C
212 RETURN
```

Zweck: *Dieses Werkzeug zeichnet ein Sprite in der Farbe, die durch den Wert von C angegeben wird.*

Anwendung: *Um dieses Werkzeug zu benutzen, müssen Sie zuerst eine Programmzeile eingeben, die SP mit der richtigen Spritenummer gleichsetzt, sodann sollten Sie eine Programmzeile eingeben, die C mit dem entsprechenden Farbcode gleichsetzt (0- 15). Den Angaben muß der Befehl GOSUB 210 folgen, der diese Unterroutine aufruft.*

Technische Beschreibung: *Diese Unterroutine unterscheidet sich dadurch von den bisherigen, daß jedes Sprite sein eigenes Farbregister hat. Die in dem Farbregister eines Sprites gespeicherte Zahl, wird durch den aktuellen Wert von C bestimmt und steht für eine der 16 verfügbaren Spritefarben.*

Alle 8 Register sind hintereinander plaziert, so daß jedes durch Addition der Spritenummer (0-7) zur Nummer des ersten Registers (53287) gefunden werden kann.

Wenn Sie die verschiedenen Farben ausprobiert haben, ändern Sie die Farbe wieder zu Gelb (C = 7 in Zeile 1160). Als nächstes plazieren wir die Sonne mit Hilfe des Werkzeugs 220 in den Himmel. Listen Sie die Zeilen 1170 und 1180. Ändern Sie die Werte von X und Y in Zeile 1170 zu:

```
1170 X = 232: Y = 10
```

Wenn Sie das Programm starten, erscheint die Sonne wieder im Himmel. Die Unterroutine 220 plaziert den Nullpunkt des Sprites an der angegebenen X- und Y-Stelle auf dem Bildschirm. (Sie erinnern sich, daß der Nullpunkt eines Sprites das obere linke Quadrat des Entwurfsrasters ist?) Diese Unterroutine muß jedesmal eingesetzt werden, wenn Sie ein neues Sprite erstellen. Ohne die Plazierung erscheint das Sprite nicht auf dem Bildschirm. Wenn Sie aber ein Sprite plaziert haben, müssen Sie es erst neu plazieren, wenn Sie es bewegen wollen.

Ein Sprite hat ein spezielles Koordinatensystem, das im folgenden Abschnitt beschrieben ist. Für die meisten Zwecke sollten Sie den Nullpunkt des Sprites innerhalb des Bereiches 0-319 für X und 0-199 für Y plazieren.

Sie haben sicherlich bemerkt, daß Teile Ihrer Sonne auf dem Bildschirm bleiben, wenn Sie zur Textanzeige zurückkehren. Sprites unterscheiden sich von anderen Formen und müssen jeweils einzeln mit dem Befehl GOSUB 140 gelöscht werden. (Die Verwendung von RUN/STOP RESTORE, um Textanzeige zu erhalten, löscht alle Sprites vom hochauflösenden Bildschirm.)

Betätigen Sie die LEERTASTE. Listen Sie die Zeilen 220-229.

Werkzeug 220::::::::::Plazieren eines Sprites bei X,Y

```
220 REM::::::::::PLACE SPRITE SP AT X,Y
221 XX = X + 24: YY = Y + 50:Z% = XX/256
222 V = XX - Z%*256: W = 53248 + SP*2
223 WW = 53264
224 PR = ABS((PEEK(WW) AND 2^SP) <>0)
225 VV = PEEK(WW) AND (255-2^SP) OR (2^SP*Z%)
226 IF PR<>Z% THEN GOSUB 140
227 POKE W,V: POKE WW,VV: GOSUB 130
228 POKE 53249 + SP*2, YY
229 RETURN
```

Zweck: *Dieses Werkzeug plaziert den Nullpunkt eines Sprites an der angegebenen Stelle X,Y auf dem Bildschirm. Der Nullpunkt ist das obere linke Quadrat des Sprite-Entwurfsrasters. Nach Plazierung des Nullpunktes an der Stelle X,Y kann der Computer den Rest des Sprites an die entsprechende Stelle plazieren. Diese Unterroutine muß jedesmal eingesetzt werden, wenn ein Sprite definiert und eingeschaltet wird, andernfalls können Sie es nicht auf dem Bildschirm sehen.*

Anwendung: *Zuerst müssen Sie SP mit der Nummer des zu plazierenden Sprites gleichsetzen. Daran anschließend folgt eine Hauptroutinenzeile in folgender Form:*

```
X = #: Y = #: GOSUB 220
```

In dieser Zeile wird # durch eine Zahl ersetzt, die eine Koordinate auf dem Bildschirm darstellt. Für X muß eine Zahl zwischen 0 und 319, für Y eine Zahl zwischen 0 und 199 eingesetzt werden. Die auf diese Weise entstandene X,Y-Koordinate gibt an, wo der Nullpunkt des Sprites auf dem Bildschirm plaziert wird.

Technische Beschreibung: *Sie wissen nun, wie Sie Sprites auf dem Bildschirm plazieren, und daß die Angaben für X sich in dem Bereich von 0 bis 319 und für Y in dem Bereich von 0 bis 199 bewegen sollten. Ferner wissen Sie daß die X,Y-Koordinate die Stelle angibt, an der die obere linke Ecke des Sprite-Raster plaziert wird. Tatsächlich kann ein Sprite aber auch außerhalb des auf dem Bildschirm sichtbaren Bereiches plaziert werden. Sprites können sich horizontal von 0 bis 511 und vertikal von 0 bis 255 bewegen. Auf dem Bildschirm ist jedoch nur der Bereich von 24 bis 343 für X und 50 bis 229 für Y sichtbar. Der zusätzliche Platz um den sichtbaren Bereich auf dem Bildschirm ermöglicht eine gleitende Bewegung des Sprites zur einen Seite des Bildschirms hinein und zur anderen hinaus. In diesem Kapitel sprechen wir nur von den "normalen" Bildschirmparametern von 320x200, den vorherigen Kapiteln angeglichen, um den Zusammenhang zwischen den einzelnen Teilen zu gewährleisten. In dieser Unterroutine PLACE SPRITE AT X,Y werden Zahlen zu Ihren X,Y-Koordinaten addiert, so daß der Nullpunkt des Sprites (0,0) immer noch dem Nullpunkt des auf dem Bildschirm sichtbaren Bereichs (0,0) entspricht. Zeile 211 übernimmt diese Addition für Sie:*

```
221 XX = X + 24: YY = Y + 50: Z% = XX/256
```

Wenn Sie den erweiterten Bereich für Ihre Sprites benutzen wollen, können Sie "XX = X + 24: YY = Y + 50:" aus dieser Zeile löschen.

Wie schon gesagt, benutzen Sie Register, um Sprites zu plazieren. Ein Register bestimmt die X-Koordinate, ein weiteres die Y-Koordinate für die Positonierung des Sprites. Jedes Sprite hat seinen eigenen Satz Register für die X- und Y-Position. Eine Änderung der

Werte in diesen Registern ändert die Plazierung des entsprechenden Sprites.

Jede Speicherstelle kann eine Zahl zwischen 0 und 255 enthalten. Da Register nur spezielle Speicherstellen sind, gilt für sie die gleiche Begrenzung. Dies funktioniert auch hervorragend, da die mit Y angegebene Position sich zwischen 0 und 255 befinden kann. Die mit X angegebene Position wirft jedoch ein Problem auf, da sie sich zwischen 0 und 511 befinden kann. Wenn das Register ein wenig größer wäre, könnte es Zahlen über 255 aufnehmen. Um dieses Problem zu lösen, wird ein weiteres Register hinzugefügt. Jedes der acht Sprites benutzt ein Bit aus diesem Register. Sprite 0 erhält Bit 0, Sprite 1 erhält Bit 1, etc. Der Computer setzt dann voraus, daß diese Bits zum Register für die Position X addiert worden sind, um sie zu vergrößern. Mit diesem hinzugefügten Bit kann das Register für die Position X Zahlen von 0 bis 511 aufnehmen. Sie können aber nicht die ganze Zahl 511 mit POKE in das Register plazieren, sondern Sie müssen sie zweiteilen. Ein Teil gelangt in das Register für die Position X des Sprites, der andere Teil gelangt in das Register, das mit anderen Sprites geteilt wird. Wenn die Positionsangabe kleiner oder gleich 255 ist, muß das Bit in dem aufgeteilten Register gleich 0 sein. Ist die Angabe größer als 255, muß das Bit gleich 1 sein. Das Schalten des Bits auf 1 bedeutet, daß Sie 256 von der aktuellen Position subtrahieren und dieses Resultat im Register für die Position X speichern können.

```
Z% = XX/256
```

Dieser Teil der Zeile entscheidet, ob das Bit in dem aufgeteilten Register gleich 0 oder gleich 1 sein soll.

```
222 V = XX - Z% * 256: W = 53248 + SP*2
```

Zeile 222 subtrahiert 256 von der Position X, wenn das Resultat von Z% gleich 1 war. Sie findet auch die richtige Registernummer für die Position X des angegebenen Sprites.

```
223 WW = 53264
```

In Zeile 223 ist WW die Speicherstelle des Registers, das von allen acht Sprites geteilt wird.

```
224 PR = ABS((PEEK(WW) AND 2^SP)    0)
225 VV = PEEK(WW) AND (255 - 2^SP) OR (2^SP*Z%)
```

Zeile 224 sucht nach der Position X des Sprites. Liegt sie links von der imaginären Begrenzung von 255, wird PR auf 0 gesetzt. Liegt sie rechts der Begrenzung, wird PR auf 1 gesetzt. Auf diese Weise wird ermittelt, ob die Begrenzung überschritten wird.

Ein Sprite kann sich horizontal nicht so einfach bewegen, wie vertikal. Wenn Sie versuchen, ein Sprite über die X-Position 255 hinaus zu bewegen, passiert etwas seltsames. In dem Moment, wenn das Sprite die imaginäre Begrenzung überschreitet, erscheint es kurzzeitig an einer anderen Stelle auf dem Bildschirm. Dies passiert jedesmal, wenn das Sprite von einer Seite der imaginären Begrenzung zur anderen bewegt wird. Die beste Lösung für dieses Problem besteht darin, das Sprite kurz vor der Änderung der Register für die Position X auszuschalten. Auf diese Weise können Sie das kurze Aufblinken des Sprites an anderer Stelle umgehen.

*Zeile 225 kontrolliert den aktuellen Inhalt der Speicherstelle, damit andere, von diesem Register gesteuerte Sprites nicht beeinträchtigt werden. Das Bit, das das aktuelle Sprite steuert, wird durch "AND (255-2^SP)" auf 0 gesetzt. Diese Technik haben wir schon bei den Beschreibungen der Unterroutinen TURN ON SPRITE SP und TURN OFF SPRITE SP behandelt. Das "OR (2^SP*Z%) gleicht der zweiten bei diesen Unterroutinen beschriebenen Befehlsformen, die das Bit einschaltet. Das Bit wird jedoch nur eingeschaltet, wenn Z% gleich 1 ist (d.h. die Position von X größer als 255 ist). Die Multiplikation von 2^SP mit Z% gibt an, ob das Bit eingeschaltet wird oder nicht.*

```
226 IF PR    Z% THEN GOSUB 140
```

In dieser Zeile gibt Z% an, auf welcher Seite der Begrenzung sich das Sprite befindet. Ist Z% gleich 0, befindet es sich links von der Begrenzung (0-255). Ist Z% gleich 1, befindet es sich rechts von der Begrenzung (256-511). Die Variable PR speichert die aktuelle Position des Sprite. Wenn dieser Bereich nicht demjenigen entspricht, in den es sich hineinbewegt, schalten Sie das Sprite für einen Moment mithilfe des Befehls GOSUB 140 aus.

```
227 POKE W,V: POKE WW, VV: GOSUB 130
```

Zeile 227 ändert die Position von X und schaltet das Sprite mit GOSUB 130 wieder ein.

```
228 POKE 53249 + SP*2,YY
```

Schließlich ändert Zeile 228 den Wert der Position von Y zur neuen Position.

Probieren Sie eine andere Plazierung für Ihr Sprite aus. Ändern Sie die Zeile 1170, so daß X gleich 319 und Y gleich 10 ist. Starten Sie das Programm.

Dieses Mal wird das Sprite fast völlig außerhalb des Bildschirms plaziert. Nur eine dünne Linie der linken Pixels bleibt auf dem Bildschirm, der Rest des Sprites hingegen ist nicht sichtbar.

Sie erinnern sich, daß der Nullpunkt die Plazierung des Sprites bestimmt? Der Nullpunkt wurde so weit rechts plaziert, daß der Rest des Sprites nicht mehr auf dem Bildschirm erscheint. Tatsächlich wurde der Rest des Sprites vom Computer einfach "vergessen", da er keinen Platz dafür hatte. Betätigen Sie die LEERTASTE.

Setzen Sie 315 für X und 10 für Y in Zeile 1170 ein und starten Sie das Programm wieder. Nun sehen Sie etwas mehr von der linken Seite der Sonne. Betätigen Sie wieder die LEERTASTE und probieren Sie diese Werte aus: X = 0 und Y = 199. Starten Sie das Programm. Das Sprite wird außerhalb des Bildschirms plaziert, dieses Mal aufgrund der Y-Koordinate. Der Nullpunkt wurde bei Y = 199 plaziert (die letzte Pixelreihe auf dem Bildschirm).

Betätigen Sie die LEERTASTE. Plazieren Sie den Nullpunkt des Sprites bei 0,195 (X = 0, Y = 195). Wenn Sie das Programm starten, sehen Sie einen schmalen Streifen der Sonne über der Umrandung in der unteren linken Ecke. Betätigen Sie die LEERTASTE. Positionieren Sie zum Schluß das Sprite bei X = 0 und Y = 0 und starten Sie das Programm. Die Sonne erscheint in der oberen linken Ecke. Der Nullpunkt liegt nun bei 0,0.

Vielleicht wollen Sie mit den Eigenschaften der Sprites, die Sie nun kennen, noch etwas experimentieren. Erweitern und reduzieren Sie die Höhe und Breite des Sprites. Ändern Sie die Farbe des Sprites. Geben Sie dem Sprite Vorrang vor den Formen. Plazieren Sie das Sprite an verschiedenen Stellen, auf dem und außerhalb des Bildschirms. Wenn Sie fortfahren wollen, vergewissern Sie sich, daß folgende Zeilen wieder in dieser Form in Ihrem Programm stehen:

```
1110 SP = 0: GOSUB 120
1120 GOSUB 130
1130 GOSUB 160
1140 GOSUB 180
1150 GOSUB 190
1160 C = 7: GOSUB 210
1170 X = 0: Y = 0
1180 GOSUB 220
```

Bewegung des Sonnen-Sprites

Wir kommen nun zu dem Abschnitt, auf den Sie sicher schon die ganze Zeit gewartet haben - die Bewegung des Sprites. Sie werden die Sonne über den Himmel bewegen, von links nach rechts. Zuerst ändern Sie die Zeilen 1170 und 1180 folgendermaßen:

```
1170 X1 = 0: Y1 = 10: X2 = 319: Y2 = 10: SD = 5
1180 GOSUB 230: GOTO 1180
```

Bevor Sie das Programm starten, geben Sie die Unterroutine MOVE SPRITE ein, die Sie unten sehen. Achten Sie darauf, daß Sie X1 und Y1 in den Zeilen 231 und 234 und den Buchstaben I hinter X und Y in den Zeilen 233 und 237 eingeben.

```
230 REM:::::::::MOVE SPRITE FROM X1,Y1
    TO X2,Y2
231 DX = X2 - X1: DY = Y2 - Y1
232 L = ABS(DX): IF ABS(DY) > L THEN L =
    ABS(DY)
233 IF L > 0 THEN XI = DX/L: YI = DY/L
234 X = X1 + .5: Y = Y1 + .5: SD = SD +
    ABS(SD = 0)
235 FOR I = 0 TO L STEP SD
236 GOSUB 220
237 X = X + XI*SD: Y = Y + YI*SD
238 NEXT I
239 RETURN
```

Sehen Sie sich Ihre neuen Programmzeilen noch einmal an. Wenn Sie glauben, daß sie richtig sind, starten Sie das Programm. Sie sehen die Sonne in einer gleichmäßigen Bewegung über den Himmel wandern. Sie erscheint auf der linken Seite, bewegt sich nach rechts über den Bildschirm und verschwindet dann aus dem Bildschirm. Wenn Sie noch einen Moment warten, erscheint die Sonne wieder auf der linken Seite und die Bewegung wird wiederholt.

Um die Bewegung zu stoppen, betätigen Sie RUN/STOP RESTORE. Mit der Betätigung der LEERTASTE erhalten Sie hier die Textanzeige nicht wieder.

```
1170 X1 = 0: Y1 = 10: X2 = 319: Y2 = 10: SD = 5
1180 GOSUB 230: GOTO 1180
```

Die Sonne beginnt ihre Bewegung an der Position von X1,Y1 (0,10), an der ihr Nullpunkt plaziert wurde. Die Bewegung setzt sich in einem geraden Weg zur Position von X2,Y2 (319,10) fort. Es ist der Nullpunkt des Sprites (obere linke Ecke des Entwurfsrasters), der dieser geraden Linie von X1,Y1 zu X2,Y2 folgt. Der Rest des Sprites wird dem Nullpunkt entsprechend ebenfalls bewegt. Ein Sprite kann horizontal, vertikal oder diagonal bewegt werden.

Am Ende der Zeile 1170 bestimmt SD = 5 die Geschwindigkeit, mit der das Sprite sich bewegt. Die Zahl 5 gibt an, wieviele Pixels auf dem geraden Weg von X1,Y1 zu X2,Y2 übersprungen werden. Je höher die Zahl ist, desto seltener muß der Computer das Sprite zeichnen, und desto schneller kann das Sprite zu seinem Ziel bewegt werden. Lassen Sie sich davon jedoch nicht zu sehr hinreißen, denn zu hohe Zahlen (Geschwindigkeiten) lassen die Bewegungen ruckartig erscheinen.

Zeile 1180 ruft die Unterroutine in Zeile 230 auf. Werkzeug 230 bewegt das zuletzt im Programm bezeichnete Sprite (SP = ?) entlang des Weges, der durch die zuletzt angegebenen Werte für X1,Y1 und X2,Y2 bestimmt wird.

Der zweite Befehl in Zeile 1180 (GOTO 1180) bewegt das Sprite immer wieder über den Bildschirm. Wenn der Computer zu Zeile 1180 gelangt, geht er zu Werkzeug 230 und bewegt das Sprite über den Bildschirm. Bei der Rückkehr zur Hauptroutine wird der Computer zurück an den Anfang der Zeile 1180 geschickt. Anschließend wird er wieder zur Unterroutine in Zeile 230 geschickt und bewegt das Sprite erneut entlang des durch X1,Y1-X2,Y2 angegebenen Weges. Diese Schleife ist "endlos", d.h. sie wird solange fortgeführt, bis Sie sie von außerhalb des Programms durch RUN/STOP RESTORE stoppen. Die LEERTASTE unterbricht das Programm nicht, da der Computer nie die Gelegenheit erhält, eine Programmzeile nach 1180 zu lesen.

Werkzeug 230:::::::::: Sprite-Steuerung von X1,Y1 nach X2,Y2

```
230 REM:::::::::MOVE SPRITE FROM X1,Y1
    TO X2,Y2
231 DX = X2 - X1: DY = Y2 - Y1
232 L = ABS(DX): IF ABS(DY) > L THEN L =
    ABS(DY)
233 IF L > 0 THEN XI = DX/L: YI = DY/L
234 X = X1 + .5: Y = Y1 + .5: SD = SD +
    ABS(SD = 0)
235 FOR I = 0 TO L STEP SD
236 GOSUB 220
```

```
237 X = X + XI*SD: Y = Y + YI*SD
238 NEXT I
239 RETURN
```

Zweck: *Dieses Werkzeug bewegt ein Sprite auf einem geraden Weg von X1,Y1 zu X2,Y2. Der Weg kann horizontal, vertikal oder diagonal verlaufen. Es ist der Nullpunkt des Sprites, der dem Weg folgt. Die Geschwindigkeit, mit der das Sprite wandert, wird durch Bestimmung der Variablen SD gesteuert. Wir schlagen vor, die Geschwindigkeit zwischen 1 und 5 anzugeben.*

Anwendung:

1. *Setzen Sie SP mit der Nummer des zu bewegenden Sprites gleich (SP = ?).*
2. *Geben Sie den Anfang (X1 = ?: Y1 = ?) und das Ende (X2 = ?: Y2 = ?) des Weges an, auf dem der Nullpunkt des Sprites wandern soll. Das Sprite bewegt sich in einer geraden Linie von X1,Y1 zu X2,Y2. Diese Koordinatenwerte sollten sich innerhalb des Bereiches für die X- und Y-Koordinaten befinden.*
3. *Geben Sie eine Zahl für die Geschwindigkeit ein (SD = ?). Je höher die Zahl, desto höher ist die Geschwindigkeit.*
4. *Den obigen Angaben muß der Befehl GOSUB 230 folgen.*

Technische Beschreibung: *Diese Unterroutine müßte Ihnen bekannt vorkommen, da sie weitestgehend mit der Unterroutine PLOT A LINE identisch ist. Anstatt jedoch eine Linie von X1,Y1 zu X2,Y2 einzuzeichnen, wollen Sie ein Sprite von X1,Y1 zu X2,Y2 bewegen. Dazu benutzen Sie dieselben Variablen, schicken den Computer aber zur Unterroutine PLACE A SPRITE an Stelle von PLOT A POINT. Bei der Sonne funktioniert dies hervorragend. Das Sprite bewegt sich langsam gleitend über den Himmel. Der Grad der Geschwindigkeit ist aber unzureichend, wenn das Sprite ein Raumschiff oder Rennwagen sein soll. Um Gegenstände zu beschleunigen, haben wir eine neue Variable hinzugefügt: SD für Geschwindigkeit. Ein Wert von 1 bedeutet "normale" Geschwindigkeit. Liegt der Wert von SD zwischen 0 und 1, wird das Sprite langsamer. Ist der Wert größer als 1, bewegt sich das Sprite schneller. Tatsächlich überspringt das Sprite mehrere Pixels, wenn die Geschwindigkeit ansteigt. Wird sie zu hoch, bewegt das Sprite sich ruckartig, während dies bei gemäßigten Geschwindigkeiten kaum zu bemerken ist.*

"SD = SD + ABS(SD = 0)" in Zeile 234 kontrolliert den Wert von SD. Ein Wert von Null würde bei der Benutzung der Unterroutine Probleme verursachen. Wenn SD gleich Null ist, setzt dieser Befehl die Variable gleich 1.

```
237 X  =  X + XI * SD : Y  =  YI * SD
```

Dieser Befehl ist identisch mit dem entsprechenden Befehl der Unterroutine PLOT A LINE, mit der Ausnahme, daß der Zuwachs von X (XI) und von Y (YI) mit der Geschwindigkeit multipliziert wird. Auf diese Weise überspringt das Sprite einige Schritte auf seinen Bildschirmreisen und wird so schneller.

```
235 FOR I  =  0 TO L STEP SD
```

In diesem Befehl wird als Neuheit STEP SD eingeführt, welches den Schleifenindex I jedesmal um den Wert von SD ansteigen läßt, wenn die Schleife durchlaufen wird, anstelle des normalen Zuwachses von 1. Dies ist verantwortlich für die Änderung im Zuwachs von X und Y. Die Änderung ermöglicht dem Sprite das Überspringen einiger Schritte durch Reduzierung der Wiederholungen, die für die Bewegung des Sprites von einer Stelle zu einer anderen notwendig sind.

Ändern Sie Zeile 1170 folgendermaßen und starten Sie das Programm wieder:

```
1170 X1 = 0: Y1 = 0: X2 = 319: Y2 = 199: SD = 10
```

Die Sonne bewegt sich diagonal über den Bildschirm nach unten. Die Bewegung ist jedoch ruckartig, da der Nullpunkt der Sonne zwar entlang des direktesten Weges von 0,0 zu 319,199 wandert, dieser Weg aber nicht völlig gerade verläuft - gar nicht verlaufen kann, da ein solcher zwischen den genannten Koordinaten nicht existiert. Wenn Sie ein Sprite bewegen wollen, überlegen Sie sich vorher, wo eine eingezeichnete Linie zwischen Ihren Punkten X1,Y1 und X2,Y2 erscheinen würde. Diese eingezeichnete Linie entspricht exakt dem Weg, dem der Nullpunkt Ihres Sprites folgt.

Unterbrechen Sie das Programm mit RUN/STOP RESTORE und ändern Sie Zeile 1170 zu:

```
1170 X1 = 160: Y1 = 0: X2 = 160: Y2 = 199: SD = 1
```

Starten Sie das Programm. Die Sonne bewegt sich von der oberen Mitte des Bildschirms gerade nach unten und aus dem Bildschirm hinaus. Durch das Loch in der Sonne sehen Sie die Linien des Schiffes. Betätigen Sie RUN/STOP RESTORE. Nun ändern Sie den Vorrang des Sprites. Ändern Sie Zeile 1150 zu:

```
1150 GOSUB 200
```

Erweitern Sie auch die Höhe des Sprites durch Änderung der Zeile 1140:

```
1140 GOSUB 170
```

Starten Sie das Programm. Die Sonne erscheint nun hoch und schmal. Sie bewegt sich von der oberen Mitte gerade nach unten. Die Geschwindigkeit ist dieses Mal wesentlich geringer, da Sie für SD 1 an Stelle von 5 angegeben haben. Die Sonne bewegt sich hinter anderen Formen, da Sie den Formen den Vorrang gegeben haben. Unterbrechen Sie das Programm mit RUN/STOP RESTORE. Listen Sie die Zeilen 1100-1190. Fügen Sie einige eigene Änderungen in den verschiedenen Programmzeilen ein und starten Sie das Programm wieder.

Wenn Sie dies beendet haben, ändern Sie die Zeilen 1110-1180 wieder zu:

```
1100 REM:::::::::SUN SPRITE
1110 SP = 0: GOSUB 120
1120 GOSUB 130
1130 GOSUB 150
1140 GOSUB 170
1150 GOSUB 200
1160 C = 7: GOSUB 21
1170 X1 = 0: Y1 = 10: X2 = 319: Y2 = 10: SD = 5
1180 GOSUB 230: GOTO 1180
```

Ändern Sie auch die DATA-Befehle, so daß die Sonne wieder als einheitliche Form erscheint. Sie können das Loch mit folgenden Zeilen ausfüllen:

```
2590 DATA 255, 255, 255
2600 DATA 255, 255, 255
2610 DATA 255, 255, 255
2620 DATA 255, 255, 255
2630 DATA 255, 255, 255
```

Nachdem Sie diese Änderungen vorgenommen haben, starten Sie das Programm, so daß das richtige Bild auf dem hochauflösenden Bildschirm angezeigt wird. Betätigen Sie RUN/STOP RESTORE. Sichern Sie das Programm unter "KAPITEL6". Fahren Sie nicht eher fort, bis das Programm sicher auf Diskette/Band abgespeichert ist.

Das Bild dieses Kapitels können Sie nicht sichern. Ihre Routine SAVE PICTURE ist nicht dazu geeignet, Sprites abzuspeichern, da dies andere Programmschritte erfordert als bei normalen Formen.

Um dieses Bild später wieder auf dem Bildschirm zu erhalten, laden Sie das Bild aus Kapitel 5 und starten dann das Programm dieses Kapitels (dies dauert nur einen Moment). Wie schon gesagt, können Sie dieses letzte Programm nur mit RUN/STOP RESTORE unterbrechen, da es eine "Endlos"-Schleife enthält.

Entwurfmöglichkeiten

Dieser Abschnitt zeigt Ihnen, wie Sie Sprites und deren Vorteile für Ihre künstlerische Kreativität nutzen können. Wenn Sie einige Erfahrung gesammelt haben und Ihre Phantasie spielen lassen, sind Ihnen keine Grenzen bei der Verwendung von Sprites gesetzt. Unsere Vorschläge in diesem Abschnitt bilden nur einen kleinen Teil der Einsatzmöglichkeiten von bewegten Sprites. Mit etwas Zeitaufwand können Sie diese Möglichkeiten in Ihren eigenen Entwürfen erforschen.

Bevor Sie jedoch damit beginnen, sollten Sie das Programm dieses Kapitels gesichert haben. Falls Sie dies noch nicht erledigt haben, tun Sie es jetzt.

In unserem ersten Beispiel lassen Sie die Sonne durch die Verwendung eines Schattens dreidimensional erscheinen. Ein Schatten kann durch Überschneidung zweier identisch geformter Sprites erzeugt werden. Sie verwenden dieselben DATA-Befehle, um die Form des Sprites zu kopieren. Dann werden die beiden Sprites übereinandergesetzt. Das untere Sprite wird ein wenig rechts vom oberen Sprite plaziert und in einer dunkleren Farbe gezeichnet, so daß mittels der Schattierung der Eindruck eines dreidimensionalen Sprites entsteht.

Ändern Sie die folgenden Programmzeilen, um Ihrer Sonne einen Schatten zu geben:

```
1160 C = 10:  GOSUB 210
1170 X = 232: Y = 10
1180 GOSUB 220
```

Diese Zeilen definieren das normale Sprite. Als nächstes geben Sie die folgenden Zeilen ein, um das zweite Sprite, den Schatten, hinter der Sonne zu plazieren. Sie sehen das Wort "RESTORE" in Zeile 1500. Dieser Befehl ist neu für Sie und wird später noch erläutert. Nach der Eingabe dieser Zeilen starten Sie das Programm.

```
1500 RESTORE
1510 SP = 1: GOSUB 120
1520 GOSUB 130
1530 GOSUB 150
1540 GOSUB 170
1550 GOSUB 200
1560 C = 2: GOSUB 210
1570 X = 235: Y = 10
1580 GOSUB 220
```

Ihr Bildschirm zeigt zuerst eine hellrote Sonne an. Dann erscheint die zweite, dunklere Sonne hinter der ersten. Der Vorrang dieser beiden Sprites wurde mit Ihren Spritenummern bestimmt. Da das erste, helle Sprite die Nummer 0 hat (SP = 0), hat es Vorrang vor jedem anderen Sprite.

Die obere Sonne ist hellrot, bestimmt durch C = 10 in Zeile 1160. Die untere Sonne, der Schatten, ist dunkelrot, bestimmt durch C = 2 in Zeile 1560. In diesem Beispiel ist das dunklere Rot die Grundfarbe. Eine Grundfarbe ist die reine, unveränderte Farbe. Die obere Sonne ist eine Aufhellung (Hellrot) der Grundfarbe. Eine Aufhellung ist, Sie erinnern sich, eine hellere, mit weiß gemischte Version der Grundfarbe. Betätigen Sie die LEERTASTE, um das Programm zu beenden.

Ändern Sie die Farbcodes in den Zeilen 1160 und 1560 und starten Sie das Programm wieder. Setzen Sie beispielsweise die Codes 7 in Zeile 1160 und 4 in Zeile 1560 ein. Die obere Sonne erscheint Gelb, ihr Schatten in verschwommenem Purpur. Farben wie Gelb und Purpur, Orange und Blau, Rot und Grün bilden Gegensatzpaare warmer und kalter Farben.

Beenden Sie das Programm mit der LEERTASTE.

Eine andere Farbkombination, die Sie ausprobieren könnten, wäre 7 in Zeile 1160 und 8 in Zeile 1560. Wenn Sie das Programm starten, erscheint die Sonne in gelb mit einem orangefarbenen Schatten. Farben wie Gelb und Orange sind sich sehr ähnlich, beide gehören zur Gruppe der warmen Farben. Solche ähnlichen Farben werden als analog bezeichnet. Analoge Farben sind miteinander verwandt und haben dieselbe Grundfarbe. Gelb und Orange beinhalten beide die Grundfarbe Gelb, Orange in einer Mischung mit Rot. Zwei andere

analoge Farben wären Grün und Blau, die sich ebenfalls sehr ähnlich sind. Beide gehören zur Gruppe der kalten Farben und haben dieselbe Grundfarbe Blau. Betätigen Sie die LEERTASTE, um wieder Textanzeige zu erhalten.

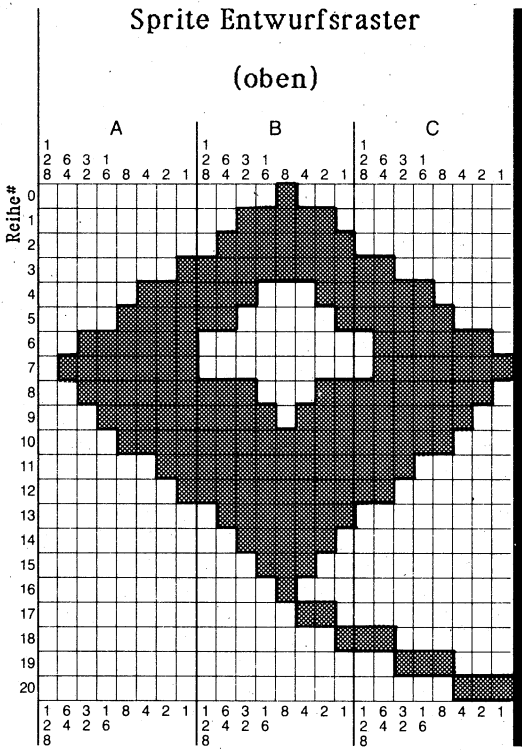
Listen Sie die Zeilen 1110-1510. Die beiden Sprites sind mit verschiedenen Spritenummern gekennzeichnet. In Zeile 1110 wird für die obere Sonne die Nummer 0 (SP = 0), in Zeile 1510 für den Schatten die Nummer 1 (SP = 1) angegeben.

Zeile 1500 enthält den BASIC-Befehl "RESTORE". Bevor der Computer diesen Befehl erreicht, hat er die DATA-Befehle für die Sonne gelesen und jede Dateneinheit "abgehakt", so daß sie während der Programmausführung nicht nochmals gelesen werden. Mit RESTORE befehlen Sie dem Computer, diese DATA-Befehle erneut zu lesen, um das zweite Sprite (Sprite 1) zu definieren. Der Computer vergißt, daß die Daten während der Programmausführung schon gelesen wurden. Auf diese Weise können dieselben Daten für das zweite Sprite wieder benutzt werden. Sie müssen jedoch immer daran denken, daß RESTORE dem Computer befiehlt, alle DATA-Befehle zu lesen. Er liest also immer den ersten DATA-Befehl in Ihrem Programm, nachdem er den Befehl RESTORE errechnet hat. Durch die Verwendung von RESTORE vergißt der Computer nicht nur den zuletzt gelesenen Satz von DATA-Befehlen.

In Zeile 1510 wird die Form des zweiten Sprites durch den Befehl GOSUB 120 definiert. Das Werkzeug 120 liest und speichert dieselben Daten für Sprite 1, die für Sprite 0 benutzt wurden.

Geben wir zur Abwechslung ein neues Sprite ein. Interessante Trickbilder entstehen durch wohlüberlegte Variationen in Ihren Sprite-Entwürfen.

Für ein neues Sprite benötigen Sie einen neuen Datenblock. Für unser Beispiel geben Sie einen Datenblock ein, der den Drachen aus der folgenden Abbildung definiert:



DATA - Befehle

BASIC-Zeile#	DATA	Summe aus A	Summe aus B	Summe aus C
1210	DATA	0	8	0
1220	DATA	0	62	0
1230	DATA	0	127	0
1240	DATA	1	255	192
1250	DATA	7	227	216
1260	DATA	15	193	248
1270	DATA	63	0	126
1280	DATA	127	0	127
1290	DATA	63	227	254
1300	DATA	31	247	252
1310	DATA	15	255	248
1320	DATA	3	255	224
1330	DATA	1	255	192
1340	DATA	0	127	0
1350	DATA	0	62	0
1360	DATA	0	28	0
1370	DATA	0	8	0
1380	DATA	0	6	0
1390	DATA	0	1	192
1400	DATA	0	0	56
1410	DATA	0	0	7

In die Mitte des Drachens haben wir ein transparentes "Fenster" gesetzt. Durch diese Öffnung können Sie alles sehen, was sich hinter dem Drachen befindet. Geben Sie folgende DATA-Befehle in Ihr Programm ein, um den Drachen zu erstellen:

```
1200 REM:::::::::KITE SPRITE DATA
1210 DATA   0,   8,   0
1220 DATA   0,  62,   0
1230 DATA   0, 127,   0
1240 DATA   1, 255, 192
1250 DATA   7, 227, 216
1260 DATA  15, 193, 248
1270 DATA  63,   0, 126
1280 DATA 127,   0, 127
1290 DATA  63, 227, 254
1300 DATA  31, 247, 252
1310 DATA  15, 255, 248
1320 DATA   3, 255, 224
```

```
1330 DATA   1, 255, 192
1340 DATA   0, 127,   0
1350 DATA   0,  62,   0
1360 DATA   0,  28,   0
1370 DATA   0,   8,   0
1380 DATA   0,   6,   0
1390 DATA   0,   1, 192
1400 DATA   0,   0,  56
1410 DATA   0,   0,   7
```

Daraufhin listen Sie die Zeilen 1100-1180. Ändern Sie diese Zeilen, so daß sie für den Drachen benutzt werden können. Schauen Sie sich die unten aufgeführten Programmzeilen an und vergleichen Sie sie mit Ihrem Programm. Sie müssen in Ihrem Programm die Zeilen 1100 und 1160 ändern. Setzen Sie in Zeile 1160 C nun mit dem Farbcode 2 gleich. Schließlich löschen Sie die Zeilen 1170 und 1180.

```
1100 REM:::::::::KITE SPRITE 0
1110 SP = 0: GOSUB 120
1120 GOSUB 130
1130 GOSUB 150
1140 GOSUB 170
1150 GOSUB 200
1160 C = 2: GOSUB 210
```

Nach der Korrektur des obigen Programmabschnittes geben Sie folgende Zeilen ein:

```
4000 SP = 0
4010 X1 = 319: Y1 = 40: X2 = 0: Y2 = 0: SD = 4
4020 GOSUB 230
4030 GOTO 4010
```

Die Zeilen 4000-4030 geben die Werte an, die für die Bewegung des Sprites mit Werkzeug 230 notwendig sind. Diese Zeilen wurden hinter die Daten für die Sonne gesetzt, so daß der Drachen sich erst bewegt, nachdem die Sonne am Himmel plaziert worden ist. Stände dieser Befehl vor Zeile 1100, würde der Drache vor einem Himmel ohne Sonne fliegen.

Löschen Sie Zeile 1500 (geben Sie 1500 ein und betätigen Sie RETURN).

Listen Sie die Zeilen 1500-1580. Diese Zeilen werden für das Sprite Sonne folgendermaßen geändert:

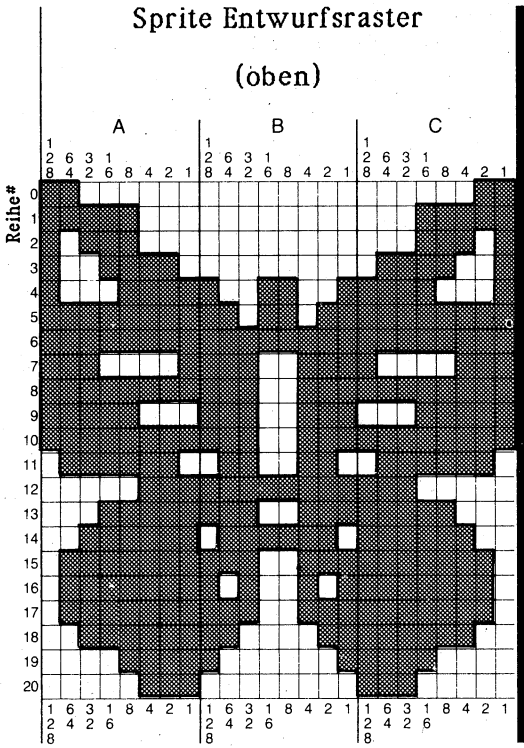
```
1500 REM:::::::::SUN SPRITE
1510 SP = 1: GOSUB 120
1520 GOSUB 130
1530 GOSUB 150
1540 GOSUB 170
1550 GOSUB 200
1560 C = 7: GOSUB 210
1570 X = 232: Y = 10
1580 GOSUB 220
```

Löschen Sie die Zeilen 1590, 1600, 1610, 1620, 1630 und 1640.

Vergewissern Sie sich, daß die Angabe des Farbcodes in Zeile 1560 zu C = 7 geändert ist. Beachten Sie, daß die Zeilen 1570 und 1580 nun ebenfalls geändert sind. Dieses Mal verharrt die Sonne an einer Stelle, sie bewegt sich nicht über den Himmel. Werkzeug 220 ist die Unterroutine, die die Sonne an einer Stelle plaziert. Da Sie zwei Sätze von Daten für die Sprites benutzen, ist der Befehl RESTORE in Zeile 1500 nicht notwendig. Starten Sie dieses Programm.

Ein roter Drache mit einer viereckigen Öffnung fliegt über den Himmel. Das Zittern des Drachens beruht auf dem diagonalen Weg, den er zurücklegt. Er beginnt seine Bewegung rechts auf dem Bildschirm (Y = 40) und wandert nach links oben (Y = 0). Die Geschwindigkeit wird mit SD = 4 bestimmt.

Jedes Sprite hat seine eigenen Daten. Diese Daten definieren die Form des Sprites. Die Öffnung innerhalb des Drachens besteht aus "transparenten" oder "leeren" Pixels. Solch eine Öffnung wird als negative Stelle des Entwurfs bezeichnet. Die negative Stelle in Ihrem Drachen hat die Form eines Diamanten, die viel besser zu Ihrem Sprite paßt, als ein einfaches Quadrat. Nehmen Sie sich beim Entwurf Ihrer Sprites für die Planung der negativen Stellen ebenso viel Zeit wie für die positiven. Gestalten Sie die negativen Stellen so interessant wie möglich. Schauen Sie sich beispielsweise die Vielfalt der negativen Stellen in der folgenden Abbildung des Schmetterlings an:



DATA - Befehle

BASIC-Zeile#	DATA	Summe aus A	Summe aus B	Summe aus C
2510	DATA	192	0	3
2520	DATA	248	0	31
2530	DATA	184	0	29
2540	DATA	158	0	121
2550	DATA	143	153	241
2560	DATA	255	219	255
2570	DATA	255	255	255
2580	DATA	225	231	135
2590	DATA	255	231	255
2600	DATA	248	231	31
2610	DATA	255	231	255
2620	DATA	126	102	126
2630	DATA	7	255	224
2640	DATA	31	231	248
2650	DATA	63	126	252
2660	DATA	127	231	254
2670	DATA	127	165	254
2680	DATA	127	231	254
2690	DATA	63	195	252
2700	DATA	15	129	240
2710	DATA	7	0	224

Bei dem Schmetterling unterscheiden sich die negativen Stellen in der Größe und der Form, wodurch ein interessantes Muster in den Flügeln entsteht.

Unterbrechen Sie die Programmausführung mit RUN/STOP RESTORE. Im nächsten Beispiel wird der Drachen direkt vor der Sonne plaziert. Dazu geben Sie neue Werte für die Plazierung und den Befehl GOSUB 220 ein. Listen Sie die Zeilen 4010-4030. Ändern Sie die Zeilen 4010 und 4020 folgendermaßen:

```
4010 X = 232: Y = 15
4020 GOSUB 220
```

Löschen Sie Zeile 4030 und starten Sie das Programm.

Dieses Mal wird der Drachen direkt vor der Sonne plaziert. Wie Sie sehen, mischen sich die Spritefarben nicht mit anderen Farbblöcken. Sie können mehrere verschieden gefärbte Sprites problemlos übereinanderschichten. Diese Technik würde sich gut für die Darstellung einer Ampel eignen. Dazu könnten Sie ein rotes, rundes Sprite

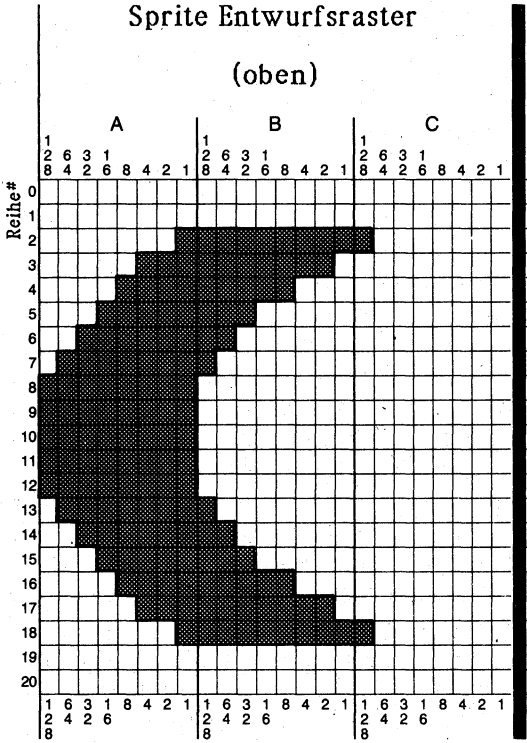
auf einem quadratischen, schwarzen Sprite plazieren. Mit den Werkzeugen 130 und 140 könnten Sie dann das Sprite ein- und ausschalten. Bei wechselndem Einsatz dieses Werkzeugs ließe sich mit Leichtigkeit eine blinkende Ampel herstellen. Probieren Sie die Technik aus, indem Sie einige Zeilen in Ihrem Programm ändern. Betätigen Sie die LEERTASTE und geben Sie diese Zeilen ein:

```
4030 GOSUB 140: GOSUB 130
4040 GOTO 4030
```

Starten Sie das Programm. Der rote Drachen erscheint nun blinkend. Dieser Effekt wird durch das Ein- und Ausschalten des Sprites erreicht. Die abwechselnde Benutzung der Befehle GOSUB 140 und GOSUB 130 bewirkt das Ein- und Ausschalten. Der Drachen erscheint, verschwindet, erscheint und verschwindet, bis Sie das Programm mit der endlosen Schleife durch RUN/STOP RESTORE beenden.

Probieren Sie einige andere Farbcodes in den Zeilen 1160 und 1560 für die beiden Sprites aus.

Wenn Sie dies beendet haben, tauschen Sie einige Daten der Sonne aus, um eine andere Form zu erhalten. Nach einigen Änderungen definieren die Daten einen Halbmond anstelle der Sonne.



DATA - Befehle

BASIC-Zeile#	DATA	Summe aus A	Summe aus B	Summe aus C
2510	DATA	0	0	0
2520	DATA	0	0	0
2530	DATA	1	255	128
2540	DATA	7	254	0
2550	DATA	15	248	0
2560	DATA	31	224	0
2570	DATA	63	192	0
2580	DATA	127	128	0
2590	DATA	255	0	0
2600	DATA	255	0	0
2610	DATA	255	0	0
2620	DATA	255	0	0
2630	DATA	255	0	0
2640	DATA	127	128	0
2650	DATA	63	192	0
2660	DATA	31	224	0
2670	DATA	15	248	0
2680	DATA	7	254	128
2690	DATA	1	255	0
2700	DATA	0	0	0
2710	DATA	0	0	0

Die folgenden Zeilen bilden den Datenblock, der den Halbmond definiert. Ändern Sie den Datenblock Ihrer Sonne, so daß er diese Dateneinheiten enthält:

```
2500 REM:::::::::MOON DATA
2510 DATA   0,   0,   0
2520 DATA   0,   0,   0
2530 DATA   1, 255, 128
2540 DATA   7, 254,   0
2550 DATA  15, 248,   0
2560 DATA  31, 224,   0
2570 DATA  63, 192,   0
2580 DATA 127, 128,   0
2590 DATA 255,   0,   0
3000 DATA 255,   0,   0
3010 DATA 255,   0,   0
3020 DATA 255,   0,   0
3030 DATA 255,   0,   0
3040 DATA 127, 128,   0
```

```
3050 DATA  63, 192,   0
3060 DATA  31, 224,   0
3070 DATA  15, 248,   0
3080 DATA   7, 254,   0
3090 DATA   1, 255, 128
4000 DATA   0,   0,   0
4010 DATA   0,   0,   0
```

Starten Sie das Programm für den Halbmond.

Probieren Sie verschiedene Formen von Sprites aus. Da ein Sprite nur in einer Farbe erscheinen kann, passen einige Formen mit Sicherheit besser als andere in dieser Beschränkung. Ansonsten aber können Sie Ihre Sprites übereinanderschichten. Verschieden gefärbte, übereinandergelagerte Schmetterlinge erzeugen z.Bsp. einen mehrfarbigen Schmetterling. Dazu plazieren Sie zuerst einen einheitlich gefärbten Schmetterling ohne Öffnung auf dem Bildschirm. Dann plazieren Sie einen deckungsgleichen Schmetterling in einer anderen Farbe und mit negativen Stellen auf dem ersten, so daß die Farbe des unteren Schmetterlings an diesen Stellen sichtbar bleibt. Die Möglichkeiten, die Ihnen die verschiedenen Eigenschaften der Sprites bieten, sind wirklich unbegrenzt.

An dieser Stelle können Sie das Programm mit Ihren Sprite-Entwürfen sichern. Im nächsten Abschnitt werden alle Eigenschaften der Sprites und die Vorgänge, mit denen Sie bewegt werden, zusammengefaßt. Am Ende finden Sie zwei abschließende Aufgaben, in denen Sie eine Wolke in das Bild dieses Kapitels setzen.

Zusammenfassung

Wir hoffen, daß Sie die Entdeckung der bunten und aufregenden Welt der Grafik auf dem **Commodore 64** genossen haben. Sie haben mit dem Einzeichnen eines einzelnen Punktes begonnen und nun ein ziemlich komplexes grafisches Kunstwerk erstellt. Obwohl nicht alle Aspekte in diesem Buch behandelt wurden, können Sie nun voller Selbstvertrauen an die Programmierung eines eigenen Bildes herangehen. Denken Sie daran, daß Sie Ihren Werkzeugkasten für alle Ihre Bilder benutzen können und sollten. Versuchen Sie außerdem, die Tips anzuwenden, die wir Ihnen in den Kapiteln zu den Einsatzmöglichkeiten der Farben gegeben haben. Sie sind vielleicht überrascht von den Ergebnissen.

Der Anhang am Ende dieses Buches enthält eine Kopie der Farbkarte, das Sprite-Entwurfsraster, sowie andere Raster, die wir in

diesem Buch benutzt haben. Sie können sich diese Raster für Ihre Skizzen kopieren. Außerdem finden Sie eine Anleitung zur Suche nach Bugs in Ihren Programmen und verschiedene zusätzliche Werkzeuge, die für jedes Grafikprogramm von Nutzen sind. Lesen Sie sich diese Abschnitte bei Gelegenheit durch.

In diesem Abschnitt sind noch einmal die verschiedenen Schritte zur Erstellung von Sprites aufgeführt. Nachfolgend erhalten Sie eine kurze Beschreibung der verschiedenen Eigenschaften von Sprites und die abschließenden Aufgaben.

Die Erzeugung eines Sprites läuft folgendermaßen ab:

1. Entwerfen Sie das Sprite auf dem "Sprite-Entwurfsraster", indem Sie den Umriß seiner Form leicht skizzieren. Dann schattieren Sie die Quadrate innerhalb dieses Umrisses. Überlegen Sie sich beim Entwurf eines Sprites, eventuell "Löcher" oder leere Stellen innerhalb der Form des Sprites einzufügen.
2. Addieren Sie die Datensummen (A,B,C) für die drei Bereiche in jeder Reihe des Entwurfs. Notieren Sie diese Summen in Zeilen neben dem "Sprite-Entwurfsraster". Ein leerer Abschnitt von 8 Quadraten ergibt eine Summe von 0. Ein vollständig schattierter Bereich von 8 Quadraten ergibt eine Summe von 255.
3. Geben Sie 21 DATA-Befehle in der Hauptroutine ein, die den Dateneinheiten in Ihrer Liste zum Entwurfsraster entsprechen.
4. Geben Sie in der Hauptroutine Ihres BASIC-Programms einen Wert (eine Zahl zwischen 0 und 7) für dieses Sprite ein. Beispielsweise bedeutet SP = 0, daß sich die Programmschritte auf das Sprite 0 beziehen. Der Vorrang der Sprites wird durch die Spritenummer bestimmt. Sprite 0 hat Vorrang vor allen anderen Sprites, Sprite 7 hat keinen Vorrang.
5. Geben Sie GOSUB 120 in der Hauptroutine ein, um die Form des Sprites im Speicher zu definieren. Das Werkzeug 120 liest und speichert die Daten in einem Array (Liste). Dieser Schritt und der obige Schritt (4) können folgendermaßen in einer Zeile eingegeben werden:

```
1110 SP = 0: GOSUB 120
```

6. Geben Sie in der Hauptroutine die ausgewählten Eigenschaften ein, die für das Sprite eingesetzt werden sollen. Die Spriteeigenschaften werden als GOSUBs eingegeben. Jedes GOSUB ruft eine bestimmte Unterroutine auf, die das Sprite beeinflußt. Unten sehen Sie mehrere Programmzeilen, die

Sprite-Eigenschaften angeben. Eine vollständige Übersicht aller Spriteeigenschaften folgt im nächsten Abschnitt.

```
1120 GOSUB 130 ("schaltet" das Sprite "ein")

1130 GOSUB 150 (erweitert das Sprite in der Breite)

1140 GOSUB 170 (erweitert das Sprite in der Höhe)

1150 GOSUB 200 (gibt einer Form Vorrang vor einem
Sprite)

1160 C = 7: GOSUB 210 (bestimmt die Farbe eines
Sprites)

1170 X = 232: Y = 10 (gibt die Plazierung für das
Sprite an)

1180 GOSUB 220 (plaziert das Sprite an der Stelle
X,Y)
```

Die Zeilen 1170 und 1180 können einfach geändert werden, um das Sprite zu bewegen. Dazu ersetzen Sie diese Zeilen durch die folgenden:

```
1170 X1 = 0: Y1 = 10: X2 = 319: Y2 = 10: SD = 5
(bestimmt die Werte für
den Weg und die Geschwindigkeit des Sprites)

1180 GOSUB 230: GOTO 1180 (bildet eine endlose
Schleife, die
das Sprite fortlaufend bewegt)
```

Zusammenfassung der Spriteeigenschaften

In diesem Abschnitt geben wir Ihnen eine Übersicht über die Eigenschaften der Sprites. Wenn Sie sich noch einmal ausführlich über jede Eigenschaft informieren wollen, wiederholen Sie den Anfang dieses Kapitels.

```
SP = 0: GOSUB 120
```

- stellt den Bezug zu Sprite 0 her. Außerdem wird die Unterroutine 120 aufgerufen, die die Form des Sprites im Speicher

definiert. Beachten Sie, daß Sie dieses Schema, um jedes Sprite zu definieren, einhalten und SP für jedes Sprite mit einer anderen Zahl (0-7) gleichsetzen müssen. Die Spritenummer bestimmt den Vorrang eines Sprites vor anderen Sprites. Sprite 0 hat größten Vorrang, Sprite 7 hat keinen Vorrang.

GOSUB 130

- "schaltet" ein Sprite "ein". Wenn das Sprite nicht auf dem Bildschirm plaziert worden ist, können Sie es nach dem Einschalten nicht sehen.

GOSUB 140

- "schaltet" das Sprite "aus", so daß es vom Bildschirm verschwindet.

GOSUB 150

- verdoppelt die Breite eines Sprites.

GOSUB 160

- reduziert die Breite eines Sprites wieder auf die ursprünglichen Ausmaße, nachdem sie mit Werkzeug 150 verdoppelt wurde.

GOSUB 170

- verdoppelt die Höhe eines Sprites.

GOSUB 180

- reduziert die Höhe eines Sprites wieder auf die ursprünglichen Ausmaße, nachdem sie mit Werkzeug 170 verdoppelt wurde.

GOSUB 190

- gibt einem Sprite Vorrang vor allen Vordergrundpixels jeder Form. Das Sprite wird vor allen Formen angezeigt, die sich an derselben Stelle wie das Sprite befinden.

GOSUB 200

- gibt den Formen in einem Bild Vorrang vor einem Sprite. Wenn das Sprite und eine Form sich an derselben Stelle auf dem Bildschirm befinden, erscheint die Form vor dem Sprite.

C = 7: GOSUB 210

- bestimmt die durch den Wert von C angegebene Farbe für das Sprite. 16 verschiedene Spritefarben (0-15) stehen Ihnen zur Verfügung. Jedes Sprite kann nur in einer Farbe erscheinen.

X = 232: Y = 10: GOSUB 220

- gibt die X- und Y-Werte für die Positionierung des Sprites auf dem Bildschirm an. Die X,Y-Pixelpunkte bestimmen die Bildschirmstelle für den Nullpunkt des Sprites. Der Rest des Sprites wird dann in Bezug zum Nullpunkt plaziert. Werkzeug 220 plaziert das Sprite nach den angegebenen X- und Y-Werten auf dem Bildschirm.

X1 = 0: Y1 = 10: X2 = 319: Y2 = 10: SD = 5

- gibt die Anfangs- und Endpunkte einer geraden Linie an, auf der das Sprite sich bewegt. X1,Y1 sind die Koordinaten des Anfangspunktes, X2,Y2 die des Endpunktes. Die Werte dieser Koordinaten bestimmen den Weg, auf dem das Sprite wandert. Ein Sprite kann sich in jeder Richtung auf einer geraden Linie bewegen. Es ist jedoch nur der Nullpunkt des Sprites, der auf dem Weg von X1,Y1 zu X2,Y2 wandert. Der Rest des Sprites bewegt sich entsprechend diesem Nullpunkt mit. SD bestimmt die Geschwindigkeit für die Bewegung des Sprites. Je höher die Zahl als Wert von SD ist, desto schneller wandert das Sprite.

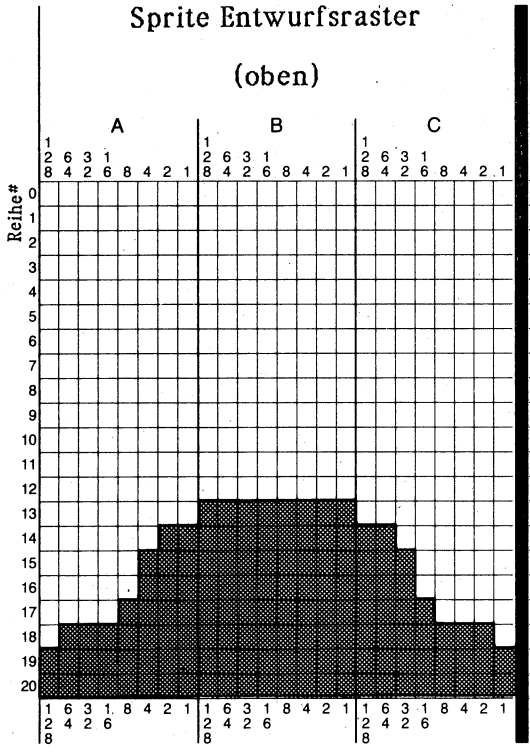
GOSUB 230

- bewegt das Sprite entlang einer geraden Linie nach den angegebenen Werten der Koordinaten X1,Y1 und X2,Y2.

Aufgabe 1

Laden Sie das Bild aus Kapitel 5 und das Programm dieses Kapitels und starten Sie es. Bei dieser Aufgabe erstellen Sie ein Sprite in

Form einer Wolke. Die folgende Abbildung zeigt das Entwurfsraster mit der Wolke. Die Datensummen - A,B,C - für jede Reihe des Sprites sind nicht mit angegeben worden.



DATA - Befehle

BASIC-Zeile#	DATA	Summe aus A	Summe aus B	Summe aus C
3210	DATA			
3220	DATA			
3230	DATA			
3240	DATA			
3250	DATA			
3260	DATA			
3270	DATA			
3280	DATA			
3290	DATA			
3300	DATA			
3310	DATA			
3320	DATA			
3330	DATA			
3340	DATA			
3350	DATA			
3360	DATA			
3370	DATA			
3380	DATA			
3390	DATA			
3400	DATA			
3410	DATA			

Sie sollen nun die Dateneinheiten für jede Reihe der Wolke herausfinden. Dann geben Sie die DATA-Befehle, die für dieses Sprite benötigt werden, in die Hauptroutine des Programms aus diesem Kapitel ein. Beginnen Sie folgendermaßen bei Zeile 3200:

```
3200 REM:::::::::CLOUD SPRITE DATA
3210 DATA    0,    0,    0
3220 DATA    0,    0,    0
```

Lösung

Die DATA-Befehle für die Wolke müssen folgendermaßen lauten:

```
3200 REM:::::::::CLOUD SPRITE DATA
3210 DATA    0,    0,    0
3220 DATA    0,    0,    0
3230 DATA    0,    0,    0
3240 DATA    0,    0,    0
3250 DATA    0,    0,    0
3260 DATA    0,    0,    0
3270 DATA    0,    0,    0
3280 DATA    0,    0,    0
3290 DATA    0,    0,    0
3300 DATA    0,    0,    0
3310 DATA    0,    0,    0
3320 DATA    0,    0,    0
3330 DATA    0,    0,    0
3340 DATA    0,  255,    0
3350 DATA    3,  255,  192
3360 DATA    7,  255,  224
3370 DATA    7,  255,  224
3380 DATA   15,  255,  240
3390 DATA  126,  255,  254
3400 DATA  255,  255,  255
3410 DATA  255,  255,  255
```

Aufgabe 2

Geben Sie, beginnend bei Zeile 3100, alle notwendigen Programmzeilen ein, um das Sprite zu definieren und folgende Eigenschaften einzusetzen:

1. Vorrang vor dem Sprite Sonne
2. erweiterte Breite
3. erweiterte Höhe
4. keinen Vorrang vor Formen
5. die Farbe weiß (Farbcode = 1)
6. Plazierung des Nullpunktes bei 215,4

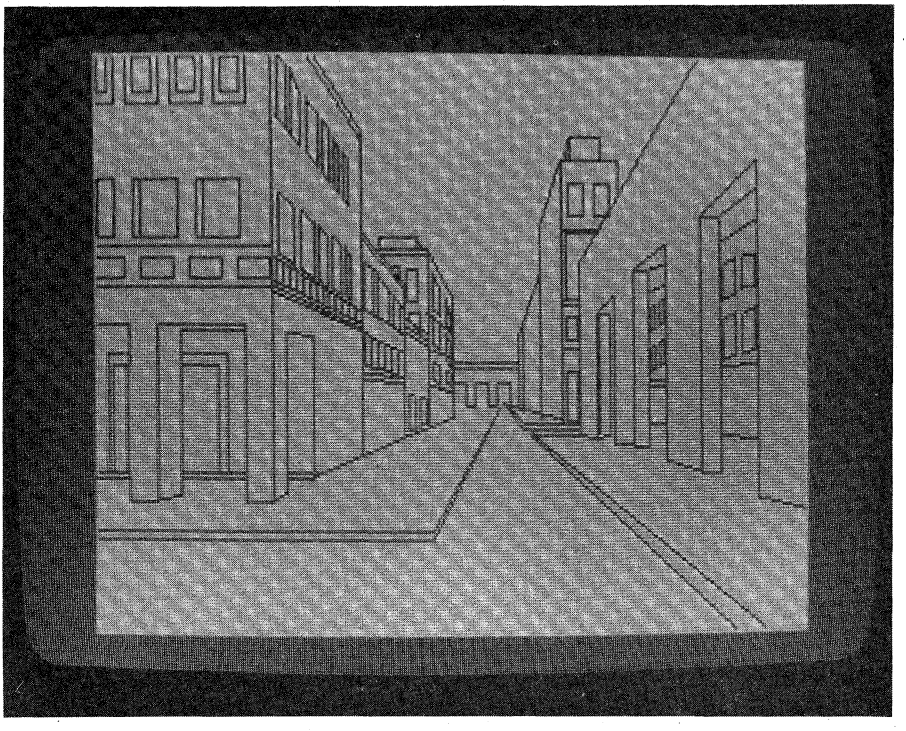
Zusätzlich ändern Sie die Eigenschaften der Sonne, so daß sie an einer Stelle bleibt (Zeile 1180 muß GOSUB 220 enthalten). Plazieren Sie den Nullpunkt der Sonne bei 232,10.

Lösung

```
1110 SP = 1: GOSUB 120
1170 X = 232: Y = 10
1180 GOSUB 220
3100 REM:::::::::CLOUD SPRITE
3110 SP = 0: GOSUB 120
3120 GOSUB 130
3130 GOSUB 150
3140 GOSUB 170
3150 GOSUB 200
3160 C = 1: GOSUB 210
3170 X = 215: Y = 4
3180 GOSUB 220
```

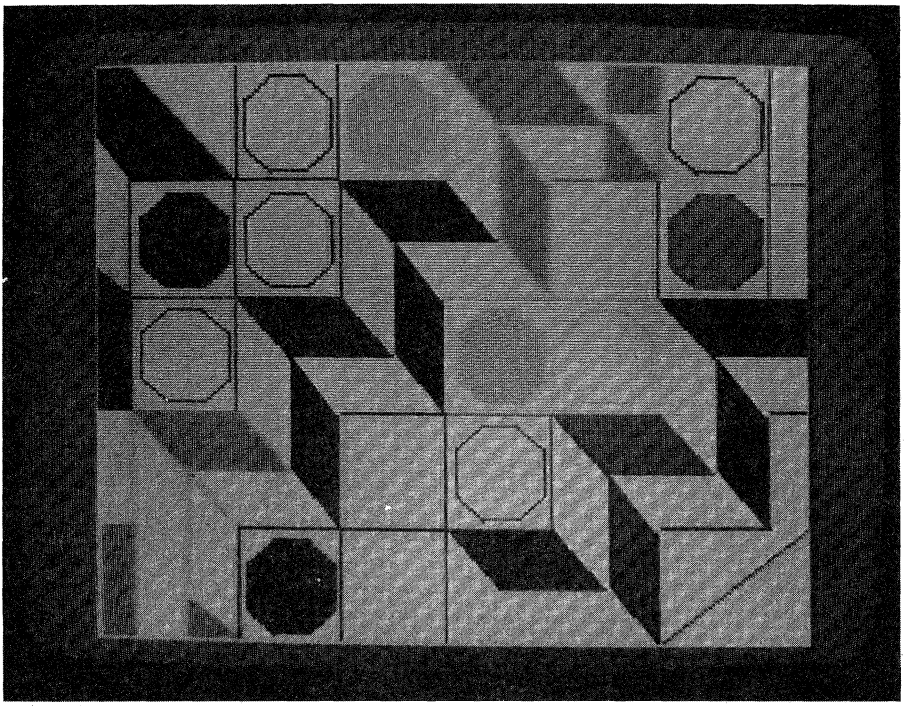
Nachwort

Da Sie nun über die Werkzeuge verfügen, mit denen Sie grafische Werke auf den Bildschirm Ihres **Commodore 64** zaubern können, bleibt es an Ihnen, die faszinierenden Möglichkeiten, die die Computergrafik bietet, zu endecken. Dieses Nachwort zeigt Ihnen als Appetitanreger drei verschiedene Bilder, die unser Künstler mit Hilfe des Werkzeugkastens aus diesem Buch entworfen hat. Das erste Motiv ist eine Stadtansicht, bei der die Unterroutine PLOT A LINE sehr wirkungsvoll eingesetzt wurde, das zweite Bild zeigt einen mit der Unterroutine DRAW A SHAPE erstellten abstrakten Entwurf. Schließlich wird im letzten Bild demonstriert, wie einige der Sprite-Werkzeuge benutzt werden können, um eine Dschungelszene zu zeichnen.



"Stadtansicht"

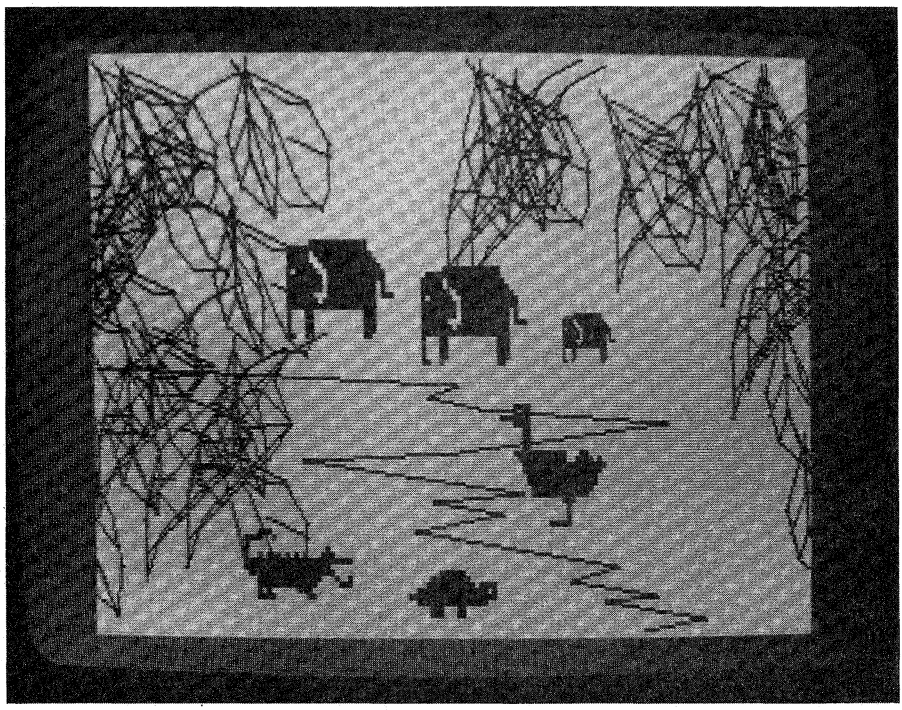
In dieser Zeichnung scheinen die Gebäude sich bis ins Unendliche fortzusetzen. Je weiter entfernt die Gebäude und Fenster liegen, desto kleiner werden sie. Der räumliche Eindruck entsteht durch die Verwendung von "Perspektive". Perspektivische Bilder zeigen Linien, die in der Ferne zu verschwinden scheinen. Der Punkt, an dem sich alle Linien treffen und verschwinden, wird als "Fluchtpunkt" bezeichnet. Sehen Sie, wie die Linien der Gebäude, Fenster und der Straße in einem Winkel zum Fluchtpunkt verlaufen. Der Verlauf der horizontalen, zusammenlaufenden Linien lenkt Ihren Blick auf diesen Punkt. Alle Linien in diesem Bild wurden mit Hilfe des Werkzeugs 80 (PLOT A LINE) gezeichnet. Zuerst zeichnete der Künstler das Bild auf dem X,Y-Pixelpunkte Raster. Alle Punkte (X,Y-Koordinaten) und die Reihenfolge, in der sie verbunden werden sollten, wurden in einer Liste notiert, die in zwei Teile geteilt wurde: der eine enthielt alle Punkte und Linien auf der linken Seite, der andere Teil alle Punkte und Linien auf der rechten Seite. Jede Liste wurde als separates Programm eingegeben und mit REM-Befehlen gekennzeichnet, um die jeweiligen Gebäude zu unterscheiden. Die Verwendung zweier Programme und mehrerer REM-Befehle waren bei der Suche nach Programmfehlern sehr hilfreich. Schließlich wurden beide Programme gestartet, um das gesamte Bild zu zeichnen, und dann mit der Unterroutine 100 als ein Bild gespeichert. In Ihren eigenen Entwürfen können Sie das Mittel der Perspektive einsetzen, um Eisenbahnschienen, Straßen oder eine Häuserzeile plastischer wirken zu lassen.



"Optische Täuschung"

In dieser abstrakten Komposition wurden geometrische Formen und Linien wiederholt eingezeichnet. Eine nicht-gegenständliche Darstellung arbeitet überwiegend mit geometrischen Formen, z.Bsp. Quadraten und Achtecken, anstelle wirklichkeitsnaher Abbildungen (Bäume, Menschen, Vögel). Um den Entwurf abwechslungsreich zu gestalten, wurden eine Sammlung von kontrastierenden Farben und verschiedene Plazierungen der Formen benutzt. Zwischen den Wiederholungen und der Vielfalt der benutzten Farben wurde ein Gleichgewicht in der Anordnung geschaffen. Achten Sie auf die Art und Weise, in der die Farbtöne den Formen zugeordnet wurden. Einige Formen sind vollständig ausgemalt, andere nur als Umriß gezeichnet. Diese Vielfalt in der Farbgebung und ihre klare Gliederung erzeugen ein interessantes Gesamtbild. Der Künstler entwarf das Bild zuerst auf dem X,Y-Pixelpunkte Raster, indem er die Umrisse zeichnete und schattierte. Die Farben wurden sorgfältig ausgewählt, so daß verschiedene Töne sich nicht in demselben Farbblock überschnitten. Für jede Form des Bildes, die wiederholt eingezeichnet werden sollte, wurde eine Liste mit den Punkten,

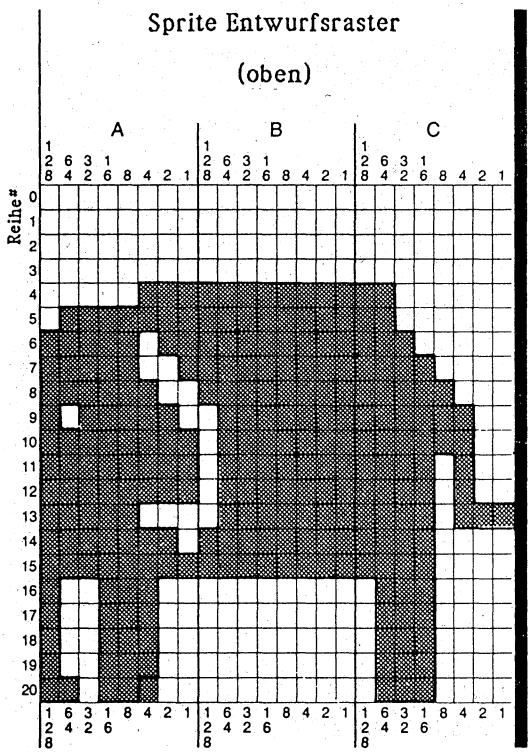
Linien und Offset-Werten zusammengestellt, in diesem Fall demnach die Oberseite der Quader, die Seite, und das Achteck. Anschließend wurde ein Programm geschrieben und in den Computer eingegeben, das diese Formen sichtbar machen sollte. Dieses Programm benutzte die Werkzeuge DRAW A SHAPE und PAINT A SHAPE. Desweiteren wurde das Programm eingegeben, welches die Formen zeichnen sollte, die nicht mehrfach vorhanden sind, wie die unregelmäßigen Formen an den Rändern des Bildes. Dieses zweite Programm zeichnete parallel die Linien zwischen den Formen ein. In dem Programm wurden die Unterroutinen 80, 90 und 110 benutzt. Schließlich wurde das vollständige Bild mit der Routine 100 gespeichert.



"Dschungelszene"

In diesem Bild wurden mehrere Blattformen verwendet, um einen Dschungel nachzubilden. Einige Bereiche sind dunkler als andere, da hier Überschneidungen der einzelnen Blattlinien auftreten. Mit wachsender Anzahl der Blätter in einem Bereich, wird auch der Farbton zunehmend dunkler und dichter. In Bereichen, in denen die Blätter

weiter voneinander entfernt plaziert sind, ist der Farbton lichter, da der helle Hintergrund an diesen Stellen durchscheint. Der Künstler benutzte drei verschiedene Blattformen. Wenn Sie sich jede Blattform in diesem Bild genau anschauen, werden Sie erkennen, daß jedes dieser drei Blätter eine einzigartige Form und Größe hat. Innerhalb der Dschungelumgebung befinden sich Sprites in Form von Tieren: drei Elephanten, ein Vogel Strauß, ein Krokodil und eine Schildkröte. Achten Sie auf die Verwendung negativer Stellen in der Form des Elephanten:

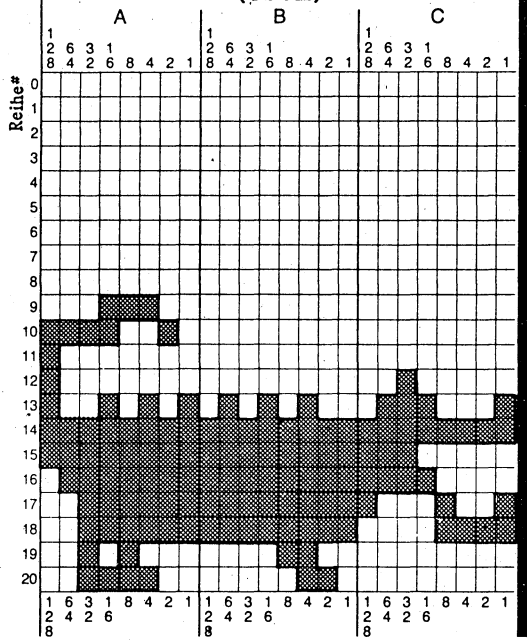


DATA - Befehle

BASIC-Zeile#	DATA	Summe aus A	Summe aus B	Summe aus C
	DATA	0	0	0
	DATA	0	0	0
	DATA	0	0	0
	DATA	0	0	0
	DATA	7	255	192
	DATA	127	255	192
	DATA	251	255	224
	DATA	249	255	240
	DATA	252	255	248
	DATA	190	127	252
	DATA	255	127	252
	DATA	255	127	244
	DATA	255	127	244
	DATA	248	127	247
	DATA	254	255	240
	DATA	255	255	240
	DATA	156	0	112
	DATA	156	0	112
	DATA	156	0	112
	DATA	156	0	112
	DATA	216	0	112

Bei diesem Sprite bilden die negativen Stellen den Umriß des Ohrs. Die nächsten Abbildungen zeigen die Entwürfe und Dateneinheiten für die anderen Tierformen:

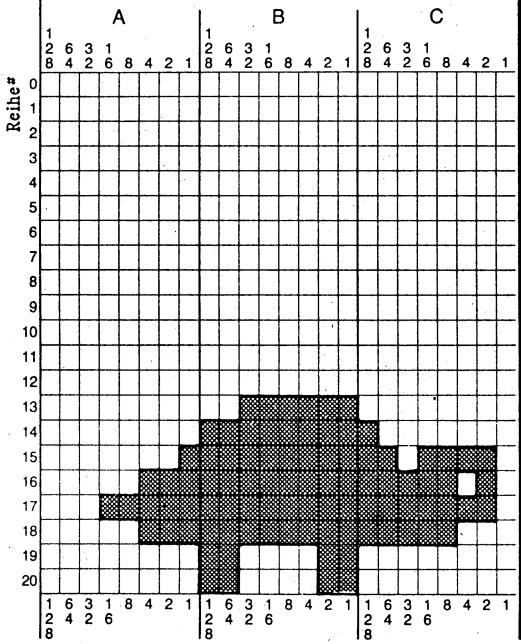
Sprite Entwurfsraster (oben)



DATA - Befehle

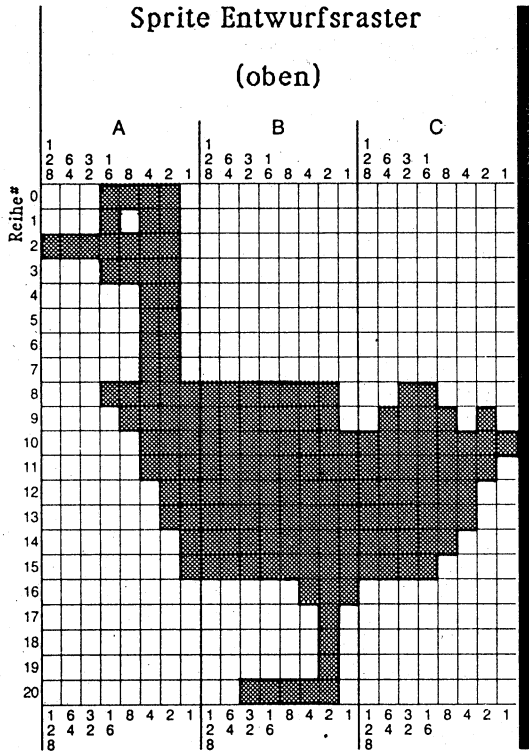
BASIC-Zeile#	DATA	Summe aus A	Summe aus B	Summe aus C
	DATA	0	0	0
	DATA	0	0	0
	DATA	0	0	0
	DATA	0	0	0
	DATA	0	0	0
	DATA	0	0	0
	DATA	0	0	0
	DATA	0	0	0
	DATA	0	0	0
	DATA	28	0	0
	DATA	242	0	0
	DATA	128	0	0
	DATA	128	0	32
	DATA	149	84	113
	DATA	255	255	255
	DATA	255	255	224
	DATA	127	255	240
	DATA	31	255	137
	DATA	63	255	15
	DATA	40	12	0
	DATA	60	6	0

Sprite Entwurfsraster (oben)



DATA - Befehle

BASIC-Zeile#	DATA	Summe aus A	Summe aus B	Summe aus C
	DATA	0	0	0
	DATA	0	0	0
	DATA	0	0	0
	DATA	0	0	0
	DATA	0	0	0
	DATA	0	0	0
	DATA	0	0	0
	DATA	0	0	0
	DATA	0	0	0
	DATA	0	0	0
	DATA	0	0	0
	DATA	0	0	0
	DATA	0	0	0
	DATA	0	0	0
	DATA	0	63	0
	DATA	0	255	128
	DATA	1	255	222
	DATA	7	255	250
	DATA	31	255	254
	DATA	7	255	248
	DATA	0	227	0
	DATA	0	227	0



DATA - Befehle

BASIC-Zeile#	DATA	Summe aus A	Summe aus B	Summe aus C
	DATA	30	0	0
	DATA	22	0	0
	DATA	254	0	0
	DATA	30	0	0
	DATA	6	0	0
	DATA	6	0	0
	DATA	6	0	0
	DATA	6	0	0
	DATA	31	254	48
	DATA	15	254	122
	DATA	7	255	255
	DATA	7	255	254
	DATA	3	255	250
	DATA	3	255	250
	DATA	1	255	248
	DATA	1	255	240
	DATA	0	7	0
	DATA	0	1	0
	DATA	0	1	0
	DATA	0	1	0
	DATA	0	62	0

Die Sprites wurden so in das Bild plaziert, daß der Eindruck eines dreidimensionalen Raumes entsteht. Beispielsweise scheinen die Elephanten weiter entfernt als der Vogel Strauß. Um diesen Effekt zu erzielen, wurden die Elephanten weiter oben auf dem Bildschirm plaziert. Für den Entwurf der Dschungelszene zeichnete der Künstler die drei Blattformen auf dem X,Y-Pixelpunkte Raster. Die Blattdaten wurden dann in Form von DATA-Befehlen in das Programm eingegeben. Diese Daten wurden in die Listen E% und L% eingelesen, um von der Unterroutine DRAW A SHAPE verarbeitet zu werden. Außerdem wurden zahlreiche Offset-Werte (X0,Y0) benutzt, um alle Blätter zu positionieren. Die Tierformen wurden, wie aus den vorherigen Abbildungen ersichtlich, einzeln auf einem Sprite-Entwurfsraster gezeichnet. Dann wurden die Spritedaten und -eigenschaften in das Programm eingegeben. Nach der Programmausführung wurde das Bild mit der Unterroutine 100 gespeichert. Da diese Unterroutine SAVE PICTURE keine Sprites speichern kann, wurde ein weiteres Programm aufgestellt, das nur die

Programmzeilen enthielt, die die Sprites betrafen. Um das Bild wieder zu sehen, wurden das Dschungelbild und das Sprite-Programm geladen und gestartet.

Anhang A
Anleitung zur Fehlersuche

Dieser Anhang bietet einen Überblick über die "Bugs" (Fehler), die häufig in Programmen auftauchen. Er besteht aus zwei Teilen: Vorbeugende Maßnahmen und Allgemeine Gegenmaßnahmen. Der erste Teil gibt hilfreiche Tips zur Vermeidung von Bugs, der zweite Teil besteht aus einer Checkliste möglicher Maßnahmen gegen häufig auftretende Programmfehler. Der Anhang ist aber keinesfalls eine vollständige und vollkommene Checkliste für sämtliche möglichen Probleme, die Sie mit Ihren Programmen haben können.

Vorbeugende Maßnahmen

Ihr Cassettenrecorder sollte möglichst weit entfernt vom Computer, Monitor und anderen metallischen Objekten stehen. Lagern Sie auch Ihre Cassetten und Disketten in sicherer Entfernung (mindestens 1,5 Meter) vom Computer. Geben Sie Ihre Programme in kleinen Buchstaben ein. Es ist erheblich einfacher, ein kleines als ein großes O (o/O) von einer Null (0) zu unterscheiden. Auch eine 8 ist besser von einem kleinen b als von einem großen B zu unterscheiden. Benennen Sie ein korrigiertes Programm lieber neu, als es mit dem Befehl "§0" unter dem alten Namen neu abzuspeichern. Dieser Befehl hat einen Fehler, der sich manchmal - nicht immer - verheerend auf Ihre gespeicherten Programme auswirken kann. Wir empfehlen Ihnen, korrigierte Programme unter neuen Namen abzuspeichern, z.B. "KAPITEL6.1", "KAPITEL6.2", etc. Erstellen Sie ein komplexes Bild in kleinen Schritten. Geben Sie einen kleinen Teil des Programms ein (ca. 2 Seiten handgeschriebenen Text) und starten Sie es. Lokalisieren und korrigieren Sie vorhandene Fehler. Die Zeit, die Sie zur Fehlersuche benötigen, kann erheblich reduziert werden, wenn Sie in dieser Weise vorgehen. Kennzeichnen Sie Ihre Programme mit mehreren REM-Befehlen, da die REMs sich später bei der Fehlersuche als sehr hilfreich erweisen. Sichern Sie Ihr Programm regelmäßig. Wenn dann tatsächlich einmal etwas Unerwartetes passiert (z.Bsp. ein Stromausfall), ist nicht Ihre ganze Arbeit verloren. Teilen Sie lange, komplizierte Programme in 2 oder 3 kleinere Programme. Dies hilft Ihnen ebenfalls bei der Fehlersuche, da Sie die Lokalisierung eines Fehlers auf einen kleineren Bereich von Programmzeilen einschränken können.

Allgemeine Gegenmaßnahmen

In Momenten tiefster Verzweiflung, wenn Sie nach einem Weg aus der Auswegslosigkeit suchen, die als der SYNTAX ERROR-Abgrund des Schreckens bekannt ist, erinnern Sie sich an folgende "Worte der Weisheit": Die Hauptursache für den Tod eines Programms ist ein Schreibfehler. Zu unser aller Glück kann jedoch ein Programm wieder zum Leben erweckt werden! Aus der Schreibfehlersuche ein Ratespiel zu machen, gibt wenig Sinn, so daß sorgfältiges Überprüfen die einzige Lösung zu einem Problem ist. Wenn ein Programm nicht funktioniert, kehren Sie zuerst zur Textanzeige zurück (benutzen Sie nicht RUN/STOP RESTORE, geben Sie sorgfältig GOSUB 30 ein und betätigen Sie RETURN). Suchen Sie nach einer Fehlermeldung mit einer Zeilennummer. Dies ist der erste und wertvollste Schlüssel zur Tilgung eines Fehlers. Wenn Sie Ihr Programm in großen Buchstaben eingegeben haben, können Sie das gesamte Listing zu kleinen Buchstaben ändern, indem Sie gleichzeitig die Tasten SHIFT und C= betätigen. Haben Sie O für 0, ein kleines l für 1 oder B für 8 eingegeben, können Sie diese schnell ermitteln. Vergewissern Sie sich, daß Sie allen notwendigen Variablen vor einem GOSUB-Befehl einen Wert zugeordnet haben. Ein komplettes Listing aller Variablen, die für jedes Werkzeug benötigt werden, finden Sie in Anhang G. Wenn ein Programm abstürzt und Ihre Hauptroutine verschwunden ist, haben Sie wahrscheinlich den Befehl GOSUB 10 (z.B. an Stelle von GOSUB 110) in Ihrer Hauptroutine eingegeben. Dies bewirkt, daß das Programm sich während seiner Ausführung selbst löscht.

Probleme beim Einzeichnen eines Punktes? Überprüfen Sie den Farbcode (C = ?) des einzuzeichnenden Punktes. Vergewissern Sie sich, daß der Code für zwei verschiedene Farben steht. Prüfen Sie, ob X und Y innerhalb der erlaubten Bereiche angegeben sind.

Probleme beim Einzeichnen einer Linie? Überprüfen Sie den Farbcode (C = ?) der einzuzeichnenden Linie. Vergewissern Sie sich, daß der Code für zwei verschiedene Farben steht. Prüfen Sie, ob Sie ein X1, ein Y1, ein X2 und ein Y2 in Ihr Programm eingegeben haben. Das Vertauschen von 1 und 2 und/oder X und Y tritt sehr häufig auf. Wenn Sie mehrere Sätze mit diesen Variablen in Ihrem Programm haben, verfolgen Sie das Programm von dem Befehl GOSUB 80 rückwärts zu den letzten X1-, Y1-, X2- und Y2-Werten. Vergleichen Sie diese mit der Linie, die Sie einzeichnen wollen.

Probleme beim Ausmalen einer Form? Prüfen Sie, ob Ihre Form nicht in Abschnitte geteilt werden muß, damit die Unterroutine sie

richtig ausmalen kann. Addieren Sie jeweils 1 zu Ihren Werten für H und W und starten Sie das Programm wieder.

Probleme beim Einzeichnen einer Form? Überprüfen Sie jede Schleife I und vergewissern Sie sich, daß Sie die Anzahl der Durchläufe richtig angegeben haben. Prüfen Sie jeden Satz DATA-Befehle. Vergewissern Sie sich, daß jeder Satz eine gerade Anzahl von Dateneinheiten enthält. Suchen Sie in den DATA-Befehlen nach Punkten ".", die an Stelle von Kommata "," eingegeben wurden. Wenn Sie die Fehlermeldung OUT OF DATA ERROR erhalten, sind Ihre DATA-Befehle nicht in Ordnung und müssen alle überprüft werden.

Probleme mit einem Sprite? Um ein Sprite auf dem Bildschirm sehen zu können, müssen Sie ihm eine Spritenummer zwischen 0 und 7 (SP = ?) zuweisen, es im Speicher definieren (GOSUB 120), es einschalten (GOSUB 130) und auf dem Bildschirm plazieren (X = ?: Y = ?: GOSUB 220).

Anhang B

Liste aller Werkzeuge

```
1 GOTO 1000
10 REM ::::::::::ZAP
11 A = 256: B = 2049: C = 1003
12 IF PEEK(B+2) + A * PEEK(B+3) >= C THEN 15
13 B = PEEK(B) + A * PEEK(B+1): ON ABS(B<>0)
   GOTO 12: END
14 A = 256: B = PEEK(251) + A * PEEK(252)
15 IF PEEK(B+1) = 0 THEN END
16 PRINT CHR$(147) PEEK(B+2) + A * PEEK(B+3):
   PRINT " GOTO 14"
17 POKE 251, B - INT(B/A) * A: POKE 252,B/A
18 POKE 631,19: POKE 632,13: POKE 633,13:
   POKE 198,3: END
20 REM::::::::::GRAPHICS
21 POKE 53265, 59
22 POKE 53272, 29
23 POKE 56576, 198
24 RETURN
30 REM::::::::::TEXT
31 POKE 53265, 27
32 POKE 53272, 21
33 POKE 56576, 199
34 RETURN
40 REM::::::::::COLORS
41 FOR I = 17408 TO 18407
42 POKE I,C
43 NEXT
44 RETURN
50 REM::::::::::PAINT BKGROUND
51 FOR I = 24576 TO 32575
52 POKE I,0
53 NEXT I
54 RETURN
60 REM::::::::::FIND A POINT
61 ROW = INT(Y/8)
62 COL = INT(X/8)
63 LINE = Y AND 7
64 BIT = 7 - (X AND 7)
65 BYTE = 24576 + ROW * 320 + COL * 8 + LINE
66 CBYTE = 17408 + ROW * 40 + COL
```

```
67 RETURN
70 REM:::::::::PLOT A POINT
71 GOSUB 60
72 POKE BYTE,PEEK(BYTE) OR 2 ^ BIT
73 POKE CBYTE,C
74 RETURN
80 REM:::::::::PLOT A LINE
81 DX = X2 - X1: DY = Y2 - Y1
82 L = ABS(DX): IF ABS(DY) > L THEN
   L = ABS(DY)
83 IF L > 0 THEN XI = DX/L: YI = DY/L
84 X = X1 + .5: Y = Y1 + .5
85 FOR I = 0 TO L
86 GOSUB 70: REM PLOT A POINT
87 X = X + XI: Y = Y + YI
88 NEXT I
89 RETURN
90 REM:::::::::PAINT A SHAPE
91 PC = PC + ABS(PC = 0): FOR X = X0 TO
   X0 + W: FL$ = "F": PR = 0
92 FOR YC = Y0 TO Y0 + H: Y = YC: GOSUB 60
93 ON ABS((PEEK(BYTE) AND 2 ^ BIT) <> 0)
   GOTO 97: IF PR = 0 THEN 96
94 PR = 0: IF FL$ = "F" THEN Y1 = YC: FL$ =
   "T": GOTO 96
95 GOSUB 99: FL$ = "F"
96 NEXT YC: GOTO 98
97 PR = 1: NEXT YC: IF FL$ = "T" THEN GOSUB 99
98 NEXT X:  RETURN
99 FOR Y = Y1 TO YC - 1: ON ABS(RND(1) < PC)
   GOSUB 70: NEXT Y: RETURN
100 REM:::::::::SAVE PICTURE
101 INPUT "ENTER FILENAME"; FILE$
102 INPUT "ENTER 8 FOR DISK, OR 1 FOR
    CASSETTE"; DE
103 SYS 57812 FILE$ + ".PIC", DE
104 POKE 174,64: POKE 175,127: POKE 193,0:
    POKE 194,96
105 SYS 62954
106 SYS 57812 FILE$ + ".COL", DE
107 POKE 174,232: POKE 175,71: POKE 193,0:
    POKE 194,68
108 SYS 62954: END
110 REM:::::::::DRAW A SHAPE
111 FOR J = 0 TO NL
112 E1 = L%(0,J): E2 = L%(1,J)
```

```
113 X1 = E%(0,E1) + X0: Y1 = E%(1,E1) + Y0
114 X2 = E%(0,E2) + X0: Y2 = E%(1,E2) + Y0
115 GOSUB 80
116 NEXT J
117 RETURN
120 REM:::::::::DEFINE SPRITE SP
121 FOR I = 0 TO 62
122 READ A
123 POKE 16384 + 64*SP + I,A
124 NEXT I
125 POKE 18424 + SP,SP
126 RETURN
130 REM:::::::::TURN ON SPRITE SP
131 POKE 53269, PEEK(53269) OR 2^SP
132 RETURN
140 REM:::::::::TURN OFF SPRITE SP
141 POKE 53269, PEEK(53269) AND (255-2^SP)
142 RETURN
150 REM:::::::::X EXPAND SPRITE SP
151 POKE 53277, PEEK(53277) OR 2^SP
152 RETURN
160 REM:::::::::X UNEXPAND SPRITE SP
161 POKE 53277, PEEK(53277) AND (255-2^SP)
162 RETURN
170 REM:::::::::Y EXPAND SPRITE SP
171 POKE 53271, PEEK(53271) OR 2^SP
172 RETURN
180 REM:::::::::Y UNEXPAND SPRITE SP
181 POKE 53271, PEEK(53271) AND (255-2^SP)
182 RETURN
190 REM:::::::::SPRITE SP PRIORITY OVER SHAPE
191 POKE 53275, PEEK(53275) AND (255-2^SP)
192 RETURN
200 REM:::::::::SHAPE PRIORITY OVER SPRITE SP
201 POKE 53275, PEEK(53275) OR 2^SP
202 RETURN
210 REM:::::::::SET SPRITE SP TO COLOR C
211 POKE 53287 + SP,C
212 RETURN
220 REM:::::::::PLACE SPRITE SP AT X,Y
221 XX = X + 24: YY = Y + 50:Z% = XX/256
222 V = XX - Z%*256: W = 53248 + SP*2
223 WW = 53264
224 PR = ABS((PEEK(WW) AND 2^SP) <>0)
225 VV = PEEK(WW) AND (255-2^SP) OR (2^SP*Z%)
226 IF PR<>Z% THEN GOSUB 140
```

```
227 POKE W,V: POKE WW,VV: GOSUB 130
228 POKE 53249 + SP*2, YY
229 RETURN
230 REM:::::::::MOVE SPRITE FROM X1,Y1
    TO X2,Y2
231 DX = X2 - X1: DY = Y2 - Y1
232 L = ABS(DX): IF ABS(DY) > L THEN L =
    ABS(DY)
233 IF L > 0 THEN XI = DX/L: YI = DY/L
234 X = X1 + .5: Y = Y1 + .5: SD = SD +
    ABS(SD = 0)
235 FOR I = 0 TO L STEP SD
236 GOSUB 220
237 X = X + XI*SD: Y = Y + YI*SD
238 NEXT I
239 RETURN
```

Anhang C
Zusätzliche Werkzeuge

Mithilfe dieses Buches haben Sie gelernt, Quadrate und Dreiecke zu zeichnen und auszumalen, verstanden als erste Schritte zum Zeichnen von beliebig verbundenen Linien und Ausmalen von Formen. Sie haben gesehen, wie diese Grundformen als Ausgangspunkt für viele bekannte Gegenstände benutzt werden könnten. Beispielsweise könnten Sie ein Quadrat für ein Haus und ein Dreieck für das Hausdach benutzen. Dieser Anhang bietet drei weitere Formen, die Sie noch oft in Ihren eigenen Zeichnungen benutzen dürften: Rechtecke, Polygone und Kreise. Wie schon die Quadrate und Dreiecke können Sie diese Formen als Grundbausteine für viele andere Gegenstände in Ihren Zeichnungen einsetzen. Bevor Sie die Werkzeuge eingeben, laden Sie das Programm des Kapitels 6 und starten die ZAP-Routine.

Werkzeug 240::::::::::Zeichnen eines Rechtecks

Diese Unterroutine kann jedes Rechteck zeichnen, wenn Sie die Höhe, Breite, Plazierung und Farbe des zu zeichnenden Rechtecks angeben. Solange Sie die Koordinaten der oberen linken Ecke kennen, müssen Sie keine weiteren Koordinaten berechnen. Geben Sie nun die Programmzeilen des neuen Werkzeugs ein:

```
240 REM::::::::::DRAW A RECTANGLE
241 X1 = X0 + W: Y1 = Y0
242 X2 = X0: Y2 = Y0: GOSUB 80
243 X1 = X0: Y1 = Y0 + H: GOSUB 80
244 X2 = X0 + W: Y2 = Y0 + H: GOSUB 80
245 X1 = X0 + W: Y1 = Y0: GOSUB 80
246 RETURN
```

Ein Beispielprogramm, das mit diesem Werkzeug ein Rechteck zeichnet, wäre:

```
1200 REM::::::::::RECTANGLE
1210 X0 = 10: Y0 = 100
1220 H = 30: W = 70
1230 C = 30: GOSUB 240
```

Die obere, linke Ecke des Rechtecks wird an der Stelle X0,Y0 plaziert, wobei es die Breite W (bestimmt durch eine bei 0 beginnende Werteskala) und die Höhe H (dito) beinhaltet. Die Umrisslinien des Rechtecks werden dabei in der durch den C-Farbwert bestimmten Farbe angezeigt.

Werkzeug 250::::::::::Zeichnen und Ausmalen eines Rechtecks

Diese Unterroutine zeichnet ein Rechteck und malt es aus, basierend auf denselben Variablen, die Sie auch in der Unterroutine 240 finden. Wenn ein Rechteck eingezeichnet und ausgemalt werden soll, ist diese Unterroutine das richtige Werkzeug dazu. Wollen Sie aber nur die Umrisse eines Rechtecks einzeichnen, benutzen Sie stattdessen Werkzeug 240. Die Zeilen der neuen Unterroutine lauten:

```
250 REM:::::::::DRAW/PAINT RECTANGLE
251 GOSUB 240
252 GOTO 90
```

Nachfolgend ein Beispielprogramm, das mit diesem Werkzeug ein Rechteck einzeichnet und ausmalt:

```
1300 REM:::::::::PAINT RECTANGLE
1310 X0 = 50: Y0 = 25
1320 W = 15: H = 10
1330 C = 46: PC = 1
1340 GOSUB 250
```

Diese Unterroutine zeichnet ein Rechteck mit Hilfe der Unterroutine 240 ein und malt es dann mit der Unterroutine 90 aus. Die obere linke Ecke des Rechtecks wird bei X0,Y0 plaziert. Das Rechteck hat eine Breite W (bestimmt durch eine bei 0 beginnenden Werteskala) und eine Höhe H (dito). Es wird mit der durch den aktuellen Wert von C angegebenen Farbe ausgefüllt. Der prozentuale Anteil der innerhalb des Rechtecks eingezeichneten Pixels entspricht dem für PC eingegebenen Dezimalbruch.

Werkzeug 260::::::::::Zeichnen eines Polygons

Ein Polygon ist eine Figur mit beliebig vielen Ecken. Obwohl mathematisch gesehen ein Polygon eine Figur mit 3 oder mehr Seiten ist, wird der Begriff allgemein verwendet, um eine Figur mit 5 oder mehr Seiten zu bezeichnen.

Mit der folgenden Unterroutine können solche Polygone gezeichnet werden:

```
260 REM:::::::::DRAW A POLYGON
261 K = 2*PI/T - .0001
262 FOR J = 0 TO 2*PI STEP K
263 W = R*SIN(J) * 1.2345
264 H = R*COS(J) * SC
265 IF J = 0 THEN X1 = X0 + W: Y1 = Y0 + H
266 X2 = X0 + W: Y2 = Y0 + H: GOSUB 80
267 X1 = X2: Y1 = Y2
268 NEXT J
269 RETURN
```

Das Zeichen für "PI", das Sie in den Zeilen 261 und 262 sehen, erhalten Sie, indem Sie gleichzeitig die SHIFT-Taste und die Taste mit dem Zeichen "^" direkt links neben RESTORE betätigen.

Wenn Sie mathematisch etwas bewandert sind, werden Ihnen einige der in den Programmzeilen aufgeführten Gleichungen verdächtig ähnlich zu jenen erscheinen, die für Kreisberechnungen verwendet werden. Eine völlig korrekte Vermutung. Die Unterroutine zeichnet eine mehrseitige Figur mit Ihren Eckpunkten auf einem imaginären Kreis. Diese Unterroutine kann direkt von der Hauptroutine oder indirekt von der Unterroutine DRAW A CIRCLE aufgerufen werden, die wir Ihnen auch noch vorstellen.

Ein Beispielprogramm, das mit dieser Unterroutine ein Polygon zeichnet, wäre:

```
1400 REM:::::::::DRAW POLYGON
1410 X0 = 100: Y0 = 75
1420 R = 20: T = 5
1430 SC = 1: C = 62
1440 GOSUB 260
```

Die Unterroutine DRAW A POLYGON zeichnet eine vieleckige Figur (in unserem Beispiel fünfeckig), indem sie einen imaginären Kreis als Grenze für die Eckpunkte benutzt. Wir beschreiben den Kreis vor dem eigentlichen Polygon.

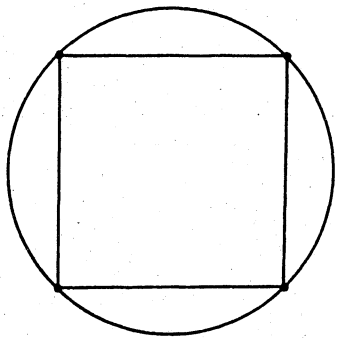
Die meisten von uns haben schon einmal einen Zirkel benutzt, um Kreise auf Papier zu zeichnen. Nach einer ähnlichen Methode zeichnet die Unterroutine DRAW A POLYGON einen imaginären Kreis auf den Monitor des Computers.

Um einen Kreis mit einem Zirkel zu zeichnen, setzen Sie die Zirkelspitze an der Stelle auf das Papier, an der sich die Mitte des

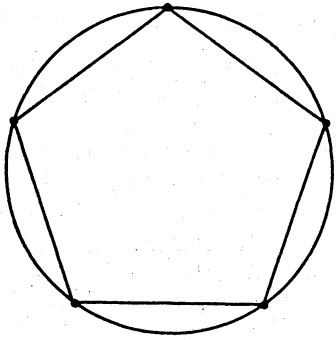
Kreises befinden soll. Dann spreizen Sie die Schreibspitze von der Zirkelspitze weg, abhängig von der gewünschten Größe des Kreises, und drehen den Zirkel um den Mittelpunkt.

Grundsätzlich führt die Unterroutine DRAW A POLYGON genau dasselbe aus. Mit der Angabe des Kreismittelpunktes und des Radius (Entfernung von der Zirkelspitze zur Schreibspitze) hat der Computer alle Informationen, die er benötigt, um den imaginären Kreis zu zeichnen.

Wenn Sie dann 4 Punkte auf diesem Kreis auswählen, um sie mit geraden Linien zu verbinden, erhalten Sie ein viereckiges Polygon:



Wählen Sie 5 Punkte aus, die Sie verbinden wollen, erhalten Sie ein fünfeckiges Polygon:

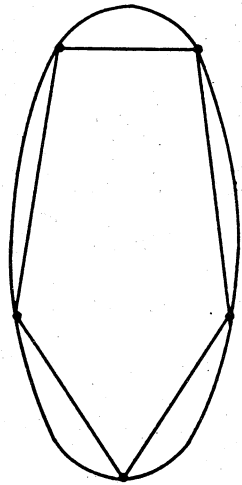


Die Unterroutine DRAW A POLYGON arbeitet prinzipiell genauso. Wenn Sie die Nummer der gewünschten Seiten (T) angeben, X0,Y0 Offset-Werte zuweisen (Mittelpunkt des Kreises) und einen Wert für

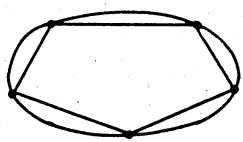
den Radius (R) eingeben, sucht der Computer sich selbst die Punkte auf dem Kreis und zeichnet Linien zwischen ihnen ein.

Zuerst wird der Größenfaktor (SC) berücksichtigt. Der Größenfaktor ist die Höhe des imaginären Kreises in Relation zu seiner Breite. Entspricht die Höhe des Kreises zweimal seiner Breite, beträgt der Größenfaktor 2 und Sie erhalten ein Oval. Entspricht die Höhe des Kreises der Hälfte seiner Breite, beträgt der Größenfaktor .5 und sie erhalten ebenfalls ein Oval.

Angenommen, der Größenfaktor beträgt 1 (ein vollkommener Kreis), dann werden die Eckpunkte des Polygons gleichmäßig verteilt, so daß alle Seiten des Vielecks dieselbe Länge haben. Wenn der Größenfaktor größer als 1 ist, wird der imaginäre Kreis in die Länge gedehnt, und das Polygon ebenso vertikal gestreckt:



Wenn der Größenfaktor kleiner als 1 ist, wird der Kreis in die Breite gedrückt, und das Polygon vertikal verkürzt:



Zeile 261 berechnet aus der angegebenen Anzahl der Seiten des Polygons und dem angegebenen Größenfaktor die Entfernung der Eckpunkte voneinander. Zeile 262 beginnt am untersten Punkt auf

dem imaginären Kreis. Dies ist immer der erste Eckpunkt des Polygons, der auf dem Bildschirm eingezeichnet wird. Das Programm durchläuft dann eine Schleife, um den "Zirkel" um den Mittelpunkt X0,Y0 zu drehen. Dabei wird ein Teil des imaginären Kreises bei jedem Schleifendurchlauf überschlagen. Jeder Punkt, an dem der "Zirkel" auf dem Kreis "auftrifft", wird als Eckpunkt des Polygons eingezeichnet. Diese Schleife wird so oft durchlaufen, bis der Anfangspunkt unten auf dem Kreis wieder erreicht ist.

Die Zeilen 263 und 264 wandeln die Eckpunkte in X,Y-Koordinaten um, damit sie von den Unterroutinen FIND A POINT und PLOT A POINT benutzt werden können. Als nächtes initialisiert Zeile 265 den ersten Eckpunkt (X1,Y1) im ersten Schleifendurchlauf. Zu diesem Zeitpunkt sind noch keine Verbindungslinien zwischen Eckpunkten gezogen, da erst ein Eckpunkt auf dem Kreis eingezeichnet worden ist (die restlichen sind bisher nur gefunden worden).

Zeile 266 berechnet den nächsten Eckpunkt (X2,Y2) und springt zur Unterroutine DRAW A LINE, um den gerade eingezeichneten Eckpunkt (X2,Y2) mit dem vorherigen (X1,Y1) zu verbinden. So entsteht die erste Linie Ihres Polygons. Zeile 267 setzt die aktuellen Eckpunkt-Koordinaten in die Variablen X1,Y1 ein, um die nächste Schleife vorzubereiten. Zeile 268 schickt den Computer zurück, um den nächsten Eckpunkt zu suchen.

Werkzeug 270::::::::::Zeichnen und Ausmalen eines Polygons

Diese Unterroutine zeichnet ein Polygon und malt es aus, basierend auf denselben Variablen, die Sie auch in der Unterroutine 260 finden. Wenn Sie ein Polygon einzeichnen und mit einer Farbe ausfüllen wollen, empfehlen wir Ihnen, diese Unterroutine zu benutzen. Wollen Sie hingegen nur die Umrisse eines Polygons einzeichnen, benutzen Sie sinnvollerweise Unterroutine 260.

Die Zeilen der Unterroutine lauten:

```
270 REM:::::::::DRAW/PAINT POLYGON
271 GOSUB 260
272 X0 = X0 - R * 1.2345
273 Y0 = Y0 - R * SC
274 H = R * 2 * SC
275 W = R * 2 * 1.2345
276 GOTO 90
```

Folgende Programmzeilen könnten Sie eingeben, um ein Polygon mit dieser Unterroutine zu zeichnen und auszumalen:

```
1500 REM:::::::::PAINT POLYGON
1510 X0 = 50: Y0 = 50
1530 R = 10: T = 6
1540 SC = 1: PC = 1: C = 78
1550 GOSUB 270
```

Die Unterroutine benutzt die Werkzeuge DRAW A POLYGON und PAINT A SHAPE, um ein Polygon einzuzeichnen und auszumalen. Umfassende Informationen zu den Variablen X0, Y0, R, T und SC finden Sie bei der obigen Erläuterung der Unterroutine 260.

PC bestimmt den prozentualen Anteil der Pixels, die innerhalb des Polygons eingezeichnet werden sollen. 1 (1.00) gibt an, daß 100% der Pixels einzuzeichnen sind. Der für C eingegebene Wert bestimmt die Farbe des Polygons.

Werkzeug 280::::::::::Zeichnen eines Kreises

Für den Computer ist ein Kreis eine der schwierigsten Figuren, da dieser eigentlich aus vielen eingezeichneten Pixels oder vielen kurzen eingezeichneten Linien besteht. Die Unterroutine DRAW A CIRCLE kann einen 30-seitigen Kreis schnell einzeichnen. Sie benutzt dabei dieselben Variablen, wie die Unterroutine 260. Geben Sie folgende Zeilen für diese Unterroutine ein:

```
280 REM:::::::::DRAW A CIRCLE
281 T = 30
282 GOTO 260
```

Mit den folgenden Programmzeilen und der oben angeführten Unterroutine könnten Sie einen Kreis einzeichnen:

```
1600 REM:::::::::DRAW A CIRCLE
1610 X0 = 100: Y0 = 100
1620 R = 15: C = 110
1630 SC = 1
1640 GOSUB 280
```

Beachten Sie, daß Sie zum Zeichnen eines Kreises dieselben Variablen benutzen, wie zum Zeichnen eines Polygons - mit der Ausnahme, daß T nicht in der Hauptroutine definiert wird. T wird in Zeile 281 der Unterroutine bestimmt. Um einen vollkommenen Kreis zu zeichnen, müssen Sie SC mit 1 gleichsetzen. Informationen zu den Variablen X0, Y0 und R finden Sie in der obigen Erläuterung der Unterroutine 260.

Werkzeug 290::::::::::Ausdruck des Bildes

Dieses letzte Werkzeug druckt eine Kopie des Bildes auf dem hochauflösenden Bildschirm mit einem VIC-1525 Drucker aus. Wenn Sie solch einen Drucker besitzen, geben Sie die folgende Unterroutine ein:

```
290 REM::::::::::PRINT PICTURE
291 OPEN 1,4: BA = 24888
292 A$ = CHR$(15) + CHR$(16) + "20" + CHR$(8)
293 FOR J = 0 TO 44: IF (JAND7) > 0 THEN BA =
    BA-8
294 BY = BA: PRINT #1, A$;
295 B1% = JAND7: B2% = 8-B1%: FORK = 0 TO 199
296 T = PEEK(BY) * 2 ^ B1% AND 127
297 B = INT(PEEK(BY + 8)/2 ^ B2%)
298 PRINT #1, CHR$(128 + T + B);
299 BY = BY + 1: IF(KAND7) = 7 THEN BY =
    BY + 312
300 NEXT K: PRINT #1: NEXT J: CLOSE 1
```

Gehen Sie folgendermaßen vor, wenn Sie diese Unterroutine benutzen wollen:

1. *Vergewissern Sie sich, daß sich ein Bild auf dem hochauflösenden Bildschirm befindet.*
2. *Schließen Sie den Drucker, wie im VIC-1525 User's Manual beschrieben, an.*
3. *Vergewissern Sie sich, daß der Drucker eingeschaltet ist, und daß sich Papier im Drucker befindet.*
4. *Geben Sie RUN 290 ein und betätigen Sie RETURN.*

Diese Unterroutine benutzt die Grafikfähigkeit des Druckers VIC-1525. Durch Teilung des Bildes auf dem hochauflösenden Bildschirm in 7x7-Pixelblöcke kann die Unterroutine den Drucker jeden Block als ein Zeichen drucken lassen. Wenn alle Blöcke ausgedruckt sind, erhalten Sie ein vollständiges Bild.

Anhang D

Beschleunigen der Werkzeuge

Obgleich die verschiedenen, in diesem Buch vorgestellten Werkzeuge jeweils wertvolle Erleichterungen beim Zeichnen auf dem **Commodore 64** bereitstellen, sind sie leider nicht immer so zeitsparend wie nützlich. Es kann zuweilen vorkommen, daß der Durchlauf eines kompletten Bildprogramms bis zu 20 Minuten in Anspruch nimmt. Ursache dafür ist, daß die Programmiersprache BASIC, die Sie in diesem Buch verwendet haben, für den Computer - genauso wie für Sie - eine Art "Fremdsprache" ist: seine genuine Sprache ist die **Maschinensprache.** Damit der Computer die von Ihnen verwendeten BASIC-Befehle in den ihm gemäßen Maschinencode umsetzen und verstehen kann, hat er einen kleinen "Übersetzer"; ein Programm also, welches BASIC liest, um es dann in Maschinensprache zu übersetzen - und das kann dauern!

Um den Vorgang zu beschleunigen, können Sie einige der zeitraubenden Werkzeuge abändern, indem Sie die Vorteile des Maschinencodes ausnutzen. Es mag Ihnen zunächst als ganz beträchtlicher (Tipp-)Aufwand erscheinen, aber er ist es tatsächlich wert. Wenn Sie die Mühe nicht scheuen, laden Sie das Programm aus Kapitel 6 und starten Sie die ZAP-Routine (RUN 10 eingeben und RETURN-Taste drücken.

Sie beginnen die Abänderungen wie folgt:

```
1 GOTO 500

41 SYS 49165,C

   (löscht Zeilen 42 und 43)

51 SYS 49157

   (löscht Zeilen 52 und 53)

71 SYS 49321,X,Y,C

   (löscht Zeilen 72 und 73)
```

81 SYS 49321,X1,Y1, TO X2,Y2,C

(löscht Zeilen 82 bis 88)

91 SYS 49551,X0,Y0,W,H,C,PC

(löscht Zeilen 92 bis 98)

99 RETURN

Im folgenden müssen Sie verschiedene DATA-Blöcke eingeben. Die DATA-Befehle speichern die Maschinencodeversionen der abgeänderten Werkzeuge.

Der erste Block ist:

```
500 FOR I = 49152 TO 49189
501 READ A: POKE I,A: T = T + A
502 NEXT I
503 IF T <> 5205 THEN PRINT "ERROR IN 500-516":
    STOP
504 T = 0
510 REM:::::::::CLEAR AND PAINT
511 DATA 134,  32,   0,   0,   0, 169,   0
512 DATA 160,  96, 162,  32, 208,   8,  32
513 DATA 241, 183, 138, 160,  68, 162,   4
514 DATA 132, 252, 160,   0, 132, 251, 145
515 DATA 251, 200, 208, 251, 230, 252, 202
516 DATA 208, 246,  96
```

Starten Sie jetzt das Programm. Entweder geschieht überhaupt nichts, oder Sie dürfen sich auf eine Fehler-(ERROR)meldung freuen. Falls diese erscheint, überprüfen Sie nochmals alle Zeilen und korrigieren alle Fehler; falls nichts auf dem Bildschirm passiert, haben Sie den betreffenden Block fehlerfrei eingegeben und können somit den nächsten in Angriff nehmen.

Nachfolgend sehen Sie 6 Programmblöcke vor sich, die Sie sukzessive, durch Starten des Programms am Ende jedes Blocks, eingeben. Gehen Sie dabei entsprechend dem Verfahren in Anfangsblock vor, wobei Sie nie mit der Eingabe eines neuen Blocks beginnen sollten, bevor nicht wirklich alle Fehler getilgt sind.

Nachdem alle Blöcke getippt und korrigiert worden sind, speichern Sie die abgeänderten Werkzeuge unter dem Dateinamen TOOL BOX ab. Die "Werkzeugkiste" wird sich im Resultat ihrer Arbeit in keinster Weise von dem im Buch ausführlich beschriebenen

unterscheiden, wohl aber in ihrer Arbeitsweise. Versuchen Sie, einfache Formen zu zeichnen und auszumalen - Sie werden positiv überrascht sein!

```
520 FOR I = 49190 TO 49263
521 READ A: POKE I,A: T = T + A
522 NEXT I
523 IF T<>8819 THEN PRINT "ERROR IN 520-541":
    STOP
524 T = 0
530 REM:::::::::FIND A POINT
531 DATA 173,  62,   3,  72,  41, 248, 168
532 DATA  32, 162, 179, 169,   0, 160, 192
533 DATA  32,  40, 186,  32, 247, 183,  24
534 DATA 173,  60,   3,  72,  41, 248, 101
535 DATA  20, 133, 251, 133, 253, 173,  61
536 DATA   3, 101,  21,  72,  74, 102, 253
537 DATA  74, 102, 253,  74, 102, 253,  24
538 DATA 105,  68, 133, 254, 104, 105,  96
539 DATA 133, 252, 104,  41,   7, 170, 104
540 DATA  41,   7, 101, 251, 144,   2, 230
541 DATA 252, 133, 251,  96
```

Eingabe beenden und Programm hier starten.

```
550 FOR I = 49264 TO 49367
551 READ A: POKE I,A: T = T + A
552 NEXT I
553 IF T<>10943 THEN PRINT "ERROR IN 550-575":
    STOP
554 T = 0
560 REM:::::::::MISC. ROUTINES
561 DATA 162,  64,  44, 162,  69,  44, 162
562 DATA  74,  44, 162,  79,  44, 162,  84
563 DATA 160,   3,  76, 212, 187, 169,  64
564 DATA  44, 169,  69,  44, 169,  84, 160
565 DATA   3,  76, 162, 187,  32, 124, 192
566 DATA  32, 247, 183, 166,  20, 164,  21
567 DATA 142,  89,   3, 140,  90,   3,  96
568 DATA 128,  64,  32,  16,   8,   4,   2
569 DATA   1,  32, 253, 174,  32, 235, 183
570 DATA 142,  62,   3, 166,  20, 164,  21
571 DATA 142,  60,   3, 140,  61,   3, 201
572 DATA 164, 240,  24,  32, 241, 183, 142
573 DATA  63,   3,  32,  38, 192, 160,   0
```

```
574 DATA 177, 251,  29, 161, 192, 145, 251
575 DATA 173,  63,   3, 145, 253,  96
```

Eingabe beenden und Programm hier starten.

```
580 FOR I = 49368 TO 49444
581 READ A:  POKE I,A: T = T + A
582 NEXT I
583 IF T<>7925 THEN PRINT "ERROR IN 580-601":
    STOP
584 T = 0
590 REM:::::::::PLOT PART 1
591 DATA  32, 115,   0,  32, 138, 173,  32
592 DATA  15, 188, 172,  60,   3, 173,  61
593 DATA   3,  32, 145, 179,  32, 112, 192
594 DATA  32,  83, 184,  32, 118, 192,  70
595 DATA 102,  32, 144, 192,  32, 241, 183
596 DATA 138, 168,  32, 162, 179,  32,  15
597 DATA 188, 172,  62,   3,  32, 162, 179
598 DATA  32, 115, 192,  32,  83, 184,  32
599 DATA 121, 192,  70, 102, 169,  84, 160
600 DATA   3,  32,  91, 188,  48,  11,  32
601 DATA  43, 188, 208,   3,  76, 192, 192
```

Eingabe beenden und Programm hier starten.

```
610 FOR I = 49445 TO 49550
611 READ A: POKE I,A: T = T + A
612 NEXT I
613 IF T<>11077 THEN PRINT "ERROR IN 610-636":
    STOP
614 T = 0
620 REM:::::::::PLOT PART 2
621 DATA  32, 144, 192,  32, 137, 192, 169
622 DATA  74, 160,   3,  32,  15, 187,  32
623 DATA 118, 192,  32, 137, 192, 169,  79
624 DATA 160,   3,  32,  15, 187,  32, 121
625 DATA 192,  32, 241, 183, 142,  63,   3
626 DATA  32, 198, 192,  32, 131, 192, 169
627 DATA  74, 160,   3,  32, 103, 184,  32
628 DATA  43, 188,  48,  52,  32, 112, 192
629 DATA  32, 247, 183, 165,  20, 166,  21
630 DATA 141,  60,   3, 142,  61,   3,  32
631 DATA 134, 192, 169,  79, 160,   3,  32
632 DATA 103, 184,  32,  43, 188,  48,  21
633 DATA  32, 115, 192,  32, 247, 183, 165
```

```
634 DATA  20, 141,  62,   3, 206,  89,   3
635 DATA 208, 191, 206,  90,   3,  16, 186
636 DATA  96
```

Eingabe beenden und Programm hier starten.

```
640 FOR I = 49551 TO 49658
641 READ A: POKE I,A: T = T + A
642 NEXT I
643 IF T<>9829 THEN PRINT "ERROR IN 640-666":
    STOP
644 T = 0
650 REM:::::::::PAINT A SHAPE PART 1
651 DATA  32,  89, 194, 141,  60,   3, 140
652 DATA  61,   3, 142,  66,   3,  32,  89
653 DATA 194, 141,  64,   3, 140,  65,   3
654 DATA 142,  67,   3,  32, 241, 183, 142
655 DATA  63,   3,  32, 253, 174,  32, 138
656 DATA 173,  32, 118, 192, 169,   0, 141
657 DATA  72,   3, 141, 73,   3, 173,  66
658 DATA   3, 141,  69,   3, 173,  67,   3
659 DATA 141,  68,   3, 173,  69,   3, 141
660 DATA  62,   3,  32,  38, 192, 160,   0
661 DATA 177, 251,  61, 161, 192, 208,  52
662 DATA 173,  73,   3, 240,  34, 169,   0
663 DATA 141,  73,   3, 173,  72,   3, 208
664 DATA  16, 173;  69,   3, 141,  70,   3
665 DATA 169,   1, 141,  71,   3, 141,  72
666 DATA   3, 208,   8
```

Eingabe beenden und Programm hier starten.

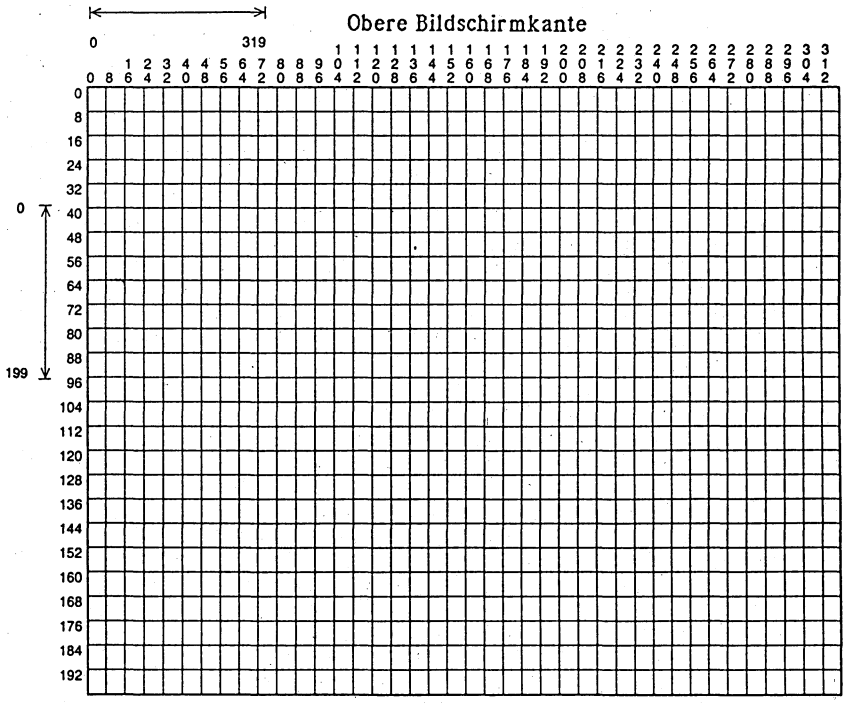
```
670 FOR I = 49659 T0 49763
671 READ A: POKE I,A: T = T + A
672 NEXT I
673 IF T<>11207 THEN PRINT "ERROR IN 670-695":
    STOP
680 REM:::::::::PAINT A SHAPE PART 2
681 DATA  32,  59, 194, 169,   0, 141,  72
682 DATA   3, 238,  69,   3, 238,  71,   3
683 DATA 206,  68,   3, 208, 188, 240,  21
684 DATA 169,   1, 141,  73,   3, 238,  69
685 DATA   3, 206,  68,   3, 208, 173, 173
686 DATA  72,   3, 240,   3,  32,  59, 194
687 DATA 238,  60,   3, 208,   3, 238,  61
688 DATA   3, 206,  64,   3, 208, 132, 206
```

```
689 DATA  65,   3,  48,   3,  76, 182, 193
690 DATA  96, 173,  70,   3, 141,  62,   3
691 DATA  32, 190, 224, 169,  74, 160,   3
692 DATA  32,  91, 188,  16,   3,  32, 198
693 DATA 192, 238,  62,   3, 206,  71,   3
694 DATA 208, 233,  96,  32, 253, 174,  32
695 DATA 235, 183, 165,  20, 164,  21,  96
```

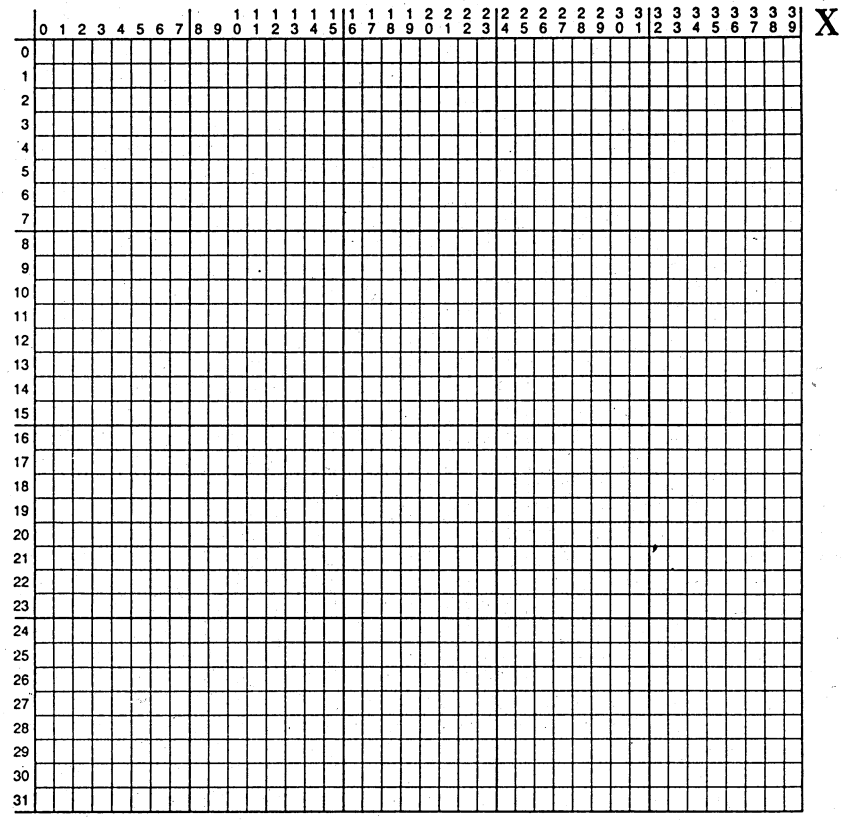
Eingabe beenden, Programm starten und - Sie haben's geschafft!

Anhang E

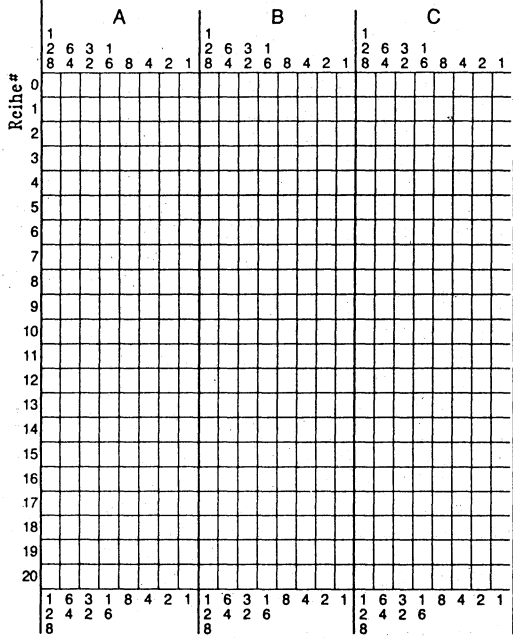
Entwurfsraster



X, Y Pixelpunkte


X
Y

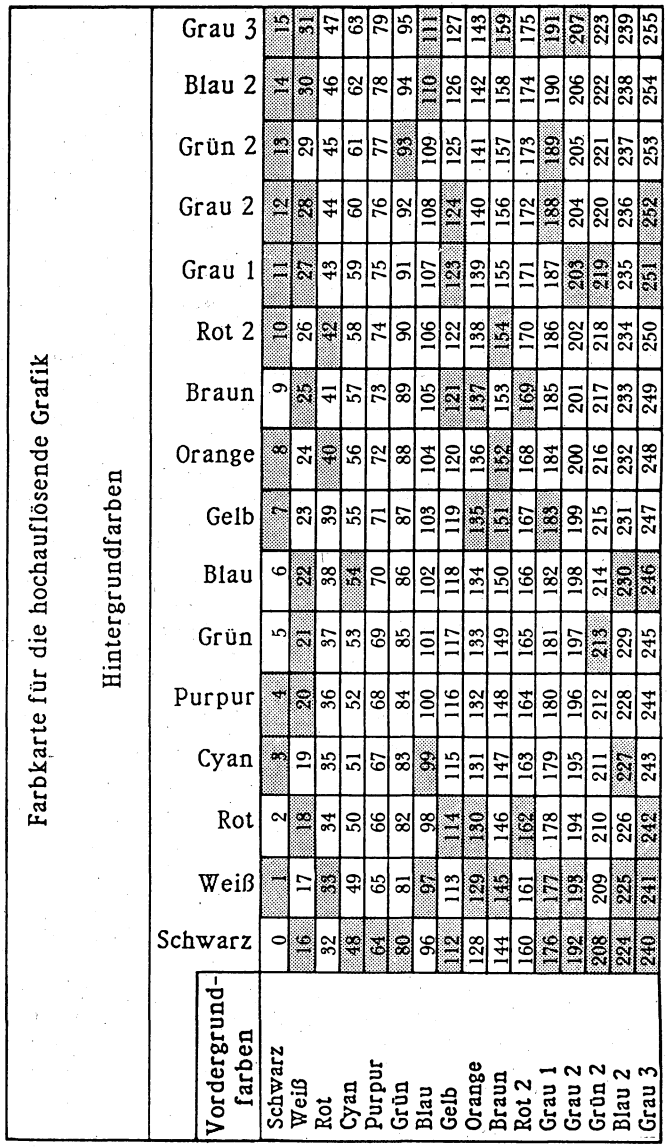
Sprite Entwurfsraster (oben)



DATA - Befehle

BASIC-Zeile#	DATA	Summe aus A	Summe aus B	Summe aus C
	DATA			
	DATA			
	DATA			
	DATA			
	DATA			
	DATA			
	DATA			
	DATA			
	DATA			
	DATA			
	DATA			
	DATA			
	DATA			
	DATA			
	DATA			
	DATA			
	DATA			
	DATA			
	DATA			
	DATA			
	DATA			

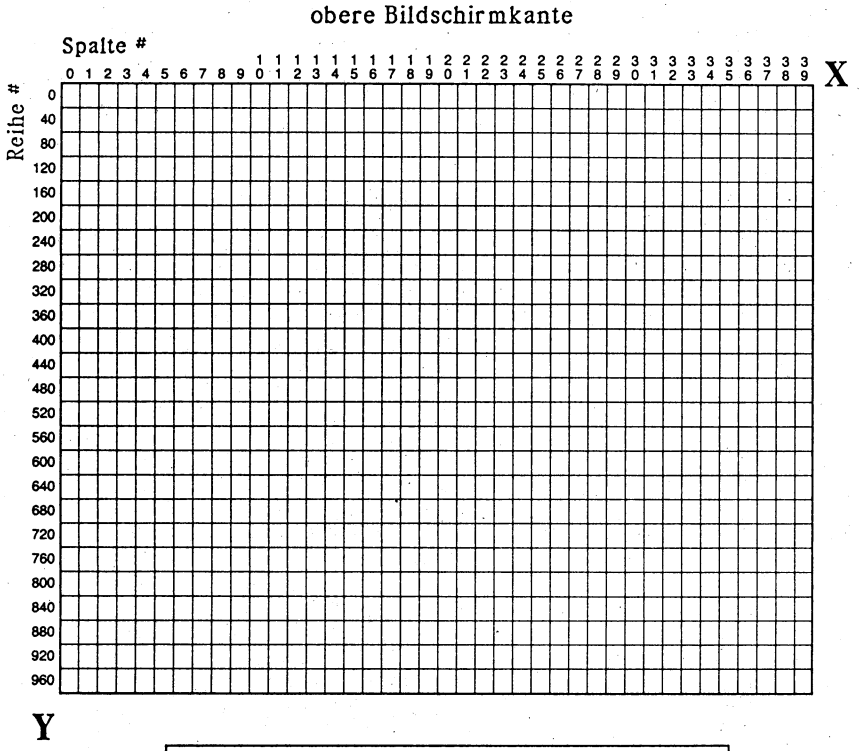
Anhang F
Farbkarte



Farbkarte für die hochauflösende Grafik

Hintergrundfarben

Vordergrundfarben	Schwarz	Weiß	Rot	Cyan	Purpur	Grün	Blau	Gelb	Orange	Braun	Rot 2	Grau 1	Grau 2	Grün 2	Blau 2	Grau 3
Schwarz	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Weiß	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Rot	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
Cyan	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
Purpur	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
Grün	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
Blau	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
Gelb	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
Orange	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
Braun	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
Rot 2	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
Grau 1	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
Grau 2	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
Grün 2	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
Blau 2	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
Grau 3	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255



Farbenblock# = Spalte# + Reihe#

Speicherstelle = 17408 + Farbenblock#

Farbenspeicher

Anhang G

Referenzliste

(Sie sehen, daß in der Spalte "Anwendung" alle Variablen mit "#" gleichgesetzt sind. Auf der Rückseite dieser Tabelle finden Sie die Bereiche für die Werte jeder Variable.)

Werkzeug #	Beschreibung	Anwendung
10	Löscht die Hauptroutine	Geben Sie RUN 10 ein und drücken Sie RETURN
20	Schaltet Grafikanzeige ein	GOSUB 20
30	Schaltet zurück auf Textanzeige	GOSUB 30
40	Bestimmt die Bildschirmfarben	C = #: GOSUB 40
50	Zeichnet den Hintergrund	GOSUB 50
60	Ermittelt einen Punkt	X=#: Y=#: GOSUB 60
70	Zeichnet einen Punkt ein	X=#: Y=#: C=#: GOSUB 70
80	Zeichnet eine Linie ein	X1=#: Y1=#: X2=#: Y2=#: C=#: GOSUB 80
90	Malt eine Form aus	X0=#: Y0=#: W=#: H=#: PC=#: C=#: GOSUB 90
100	Speichert ein Bild	geben Sie RUN 100 ein und drücken Sie RETURN geben Sie Dateinamen und die Gerätenummer (8 oder 1) ein
110	Zeichnet eine Form	E%(#,I) füllen: L%(#,I) füllen NL=#: C=#: X0=#: Y0=#: GOSUB 110

120 Definiert Sprite SP	geben Sie 63 Daten-Einheiten in DATA-Befehlen ein SP=#: GOSUB 120
130 Schaltet Sprite SP ein	SP=#: GOSUB 130
140 Schaltet Sprite SP aus	SP=#: GOSUB 140
150 Erweitert X von Sprite SP	SP=#: GOSUB 150
160 Reduziert X von Sprite SP	SP=#: GOSUB 160
170 Erweitert Y von Sprite SP	SP=#: GOSUB 170
180 Reduziert Y von Sprite SP	SP=#: GOSUB 180
190 Gibt Sprite Vorrang vor Form	SP=#: GOSUB 190
200 Gibt Form Vorrang vor Sprite	SP=#: GOSUB 200
210 Bestimmt Farbe C für Sprite SP	C=#: SP=#: GOSUB 210
220 Plaziert Sprite SP an X,Y	X=#: Y=#: SP=#: GOSUB 220
230 Bewegt Sprite SP von X1,Y1 zu X2,Y2	X1=#: Y1=#: X2=#: Y2=#: SP=#: GOSUB 230

Die folgenden Variablen werden von den Unterroutinen in diesem Buch benutzt:

Variable für die Werte	Beschreibung	Bereich
X0	Offset-Variable X für Formen	0 - 319
Y0	Offset-Variable Y für Formen	0 - 199
X1	Anfangskoordinate X	0 - 319
Y1	Anfangskoordinate Y	0 - 199
X2	Endkoordinate X	0 - 319
Y2	Endkoordinate Y	0 - 199
C	Farbcode	0 - 255 für Formen 0 - 15 für Sprites
X	X-Koordinate	0 - 319
Y	Y-Koordinate	0 - 199
W	Breite einer Form	0 - 319
H	Höhe einer Form	0 - 199
PC	% des auszumalenden Bereiches	0.0 - 1.0
NL	# der Linien in einer Form	>0
E%	Liste der Endpunkt-Daten	N/A
L%	Liste der Linien-Daten	N/A
SP	Spritenummer	0 - 7
SD	Geschwindigkeit eines Sprites	>0

(Die folgenden Variablen werden von einigen zusätzlichen Werkzeugen aus dem Anhang C benutzt:)

R	Radius einer Form	0 - 100
T	# der Seiten eines Polygons	>2
SC	Vertikale Größe einer Form	1 = normal <1 - flacher <1 - höher

MERLIN 64

von Glen Bredon

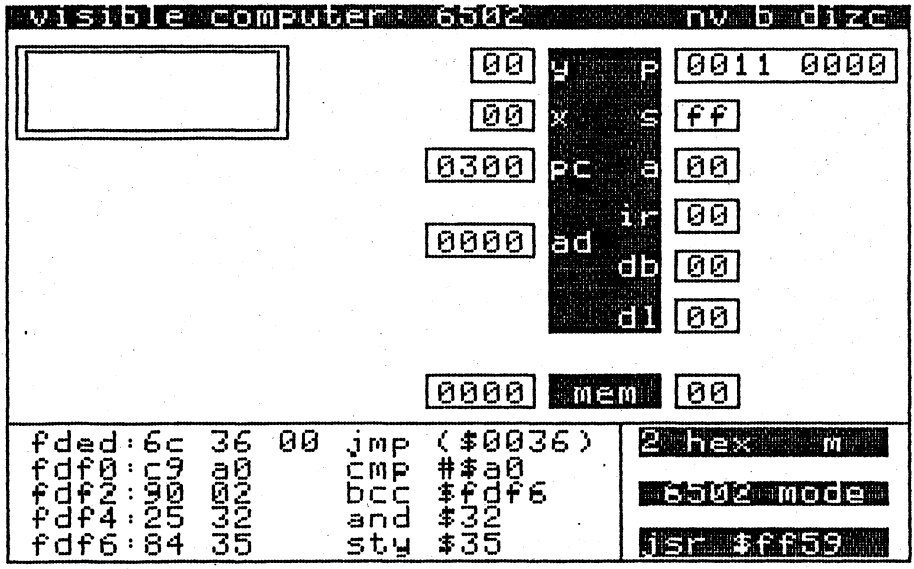
Ein professioneller Macro-Assembler für den Commodore 64!
Neben allen Standard-Features bietet MERLIN 64 u.a.:

- Einführung in die 6502 Assemblersprache
- komfortabler Editor mit globalen Such- und Ersetzfunktionen
- liest und schreibt Text- und Binärfiles
- Bibliothek mit vielen nützlichen Unterprogrammen
- Disassembler
- deutsches Handbuch

Ausführliche Informationen von:

Pandabooks
Bismarckstr. 67
D-1000 Berlin 12

The Visible Computer: 6502




```
# go
```

Ein Simulationsprogramm, das Sie in das Innere des 6502 Mikroprozessors führt. Sie sehen auf dem Bildschirm, wie die einzelnen Instruktionen in Zeitlupe ausgeführt werden, wie sich die Register und die Flags verändern. Ein unverzichtbares Hilfsmittel beim Erlernen der Assembler- Programmierung - danach ein wertvolles Werkzeug beim Testen Ihrer eigenen Programme.
Komplett mit über 30 Beispielprogrammen und deutschem Handbuch. (150 Seiten).

DM 99,--

Erhältlich in vielen Computershops, im guten Buchhandel oder direkt von:

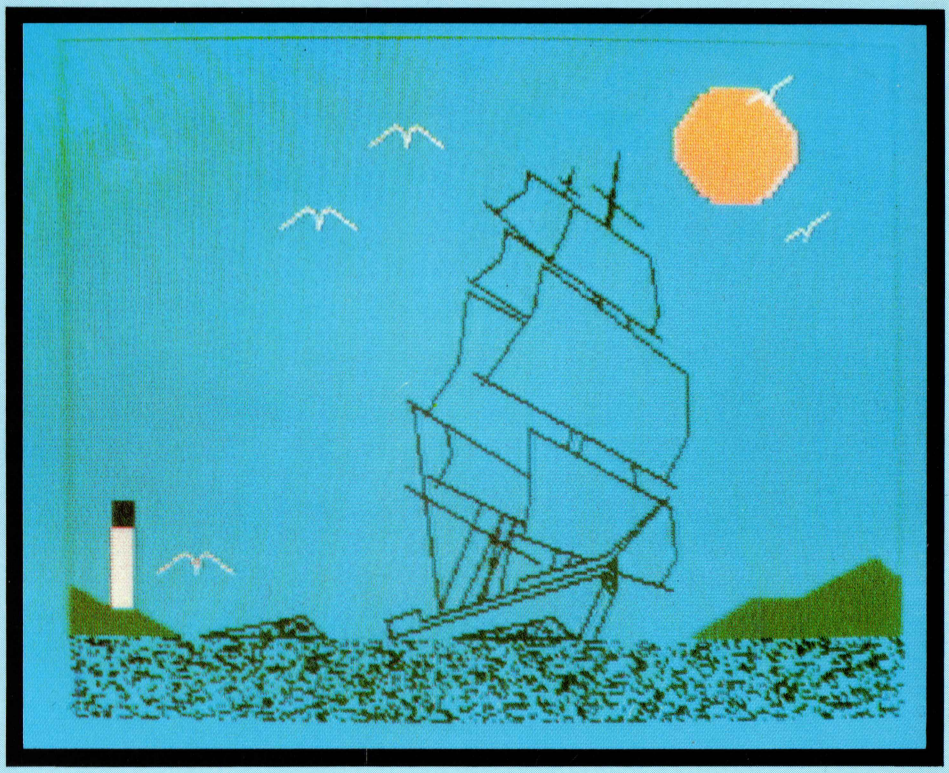
Pandabooks
Bismarckstr. 67
D-1000 Berlin 12

Von den gleichen Autoren ist erschienen:

C64 Grafik Handbuch für Fortgeschrittene

ISBN: 3-89058-016-5 DM 49,-- (komplett mit Disk DM 89,--)

Bei der Entwicklung eines kompletten Arcade-Spiels werden schrittweise die Theorie und Technik der schnellen, bewegten Grafikprogrammierung vorgestellt: Translation, Skalierung, Rotation, Maschinensprache, Unterbrechungen. Hier werden Ihnen keine fertigen Programme vorgesetzt, sondern Werkzeuge und Wissen zur Verfügung gestellt, mit denen Sie Ihre eigenen Ideen umsetzen konnen.



Einführung in die hochauflösende Farbgrafik des Commodore 64. Mit diesem Buch lernen Sie:

— den Umgang mit Vorder- und Hintergrundfarben

— das Zeichnen von Punkten und Linien

— den Aufbau und das Plazieren von Formen

— das Animieren von kleinen Objekten

Wenn die komplette Grafik erstellt ist, dann haben Sie sich »nebenbei« eine Sammlung von »Programmierwerkzeugen« zugelegt, die Sie gewinnbringend für neue Entwürfe benutzen können.

Pandabooks Berlin **ISBN: 3–89058–013–0**