Einsteigen - Verstehen - Beherrschen

DM 3,80 öS 30 sfr 3,80

computer kurs

Ein wöchentliches Sammelwerk

Computergrafik

Rechner Link 480Z

Die Nacht der Wölfe

Anweisungen und Adressen

Thermodrucker für den Spectrum

Programmierkurs PROLOG

Heft 46

computer kurs

Heft 46

Inhalt

WIE SIE JEDE WOCHE IHR HEFT BEKOMMEN

Computer Kurs ist ein wöchentlich erscheinendes Sammelwerk. Die Gesamtzahl der Hefte ergibt ein vollständiges Computer-Nachschlagewerk. Damit Sie jede Woche Ihr Heft erhalten, bitten Sie Ihren Zeitschriftenhändler, Computer Kurs für Sie zu reservieren.

Zurückliegende Hefte
Ihr Zeitschriftenhändler besorgt Ihnen gerne zurückliegende Hefte. Sie können sie aber auch direkt beim Verlag bestellen.

Deutschland: Das einzelne Heft kostet DM 3,80. Bitte füllen Sie eine Postzahlkarte aus an: Marshall Cavendish Int. Ltd. (MCI), Sammelwerk-Service, Postgiroamt Hamburg 48064-202, Postfach 105703, 2000 Hamburg 1, Kennwort: Computer Kurs

Österreich: Das einzelne Heft kostet öS 30. Bitte füllen Sie eine Zahlkarte aus an: Computer Kurs, Wollzeile 11, 1011 Wien, Postscheckkonto Wien 7857201 oder legen Sie Ihrer Bestellung einen Verrechnungsscheck bei. Kennwort: Computer Kurs.

Schweiz: Das einzelne Heft kostet sfr 3,80. Bitte wenden Sie sich an Ihren Kiosk; dort werden Sie jederzeit die gewünschten Exemplare erhalten.

Abonnement

Sie können Computer Kurs auch alle 2 Wochen (je 2 Ausgaben) per Post zum gleichen Preis im Abonnement beziehen. Der Abopreis für 12 Ausgaben beträgt DM 45,60 inkl. Mwst., den wir Ihnen nach Eingang der Bestellung berechnen. Bitte senden Sie Ihre Bestellung an: Marshall Cavendish Int. Ltd. (MCI), Sammelwerk Service, Postgiroamt Hamburg 86853-201, Postfach 105703, 2000 Hamburg 1, Kennwort: Abo Computer Kurs. Bitte geben Sie an, ab welcher Nummer das Abo beginnen soll und ob Sie regelmäßig für jeweils 12 Folgen einen Sammelordner wünschen.

WICHTIG: Bei Ihren Bestellungen muß der linke Abschnitt der Zahlkarte Ihre vollständige Adresse enthalten, damit Sie die Hefte schnell und sicher erhalten. Überweisen Sie durch Ihre Bank, so muß die Überweisungskopie Ihre vollständige Anschrift gut leserlich enthalten.

SAMMELORDNER
Sie können die Sammelordner entweder direkt bei Ihrem Zeitschriftenhändler kaufen (falls nicht vorrätig, bestellt er sie gerne für Sie) oder aber Sie bestellen die Sammelordner für den gleichen Preis beim Verlag wie folgt:

Deutschland: Der Sammelordner kostet DM 12. Bitte füllen Sie eine Zahlkarte aus an: Marshall Cavendish International Ltd. (MCI), Sammelwerk-Service, Postgiroamt Hamburg 48064-202, Postfach 105703, 2000 Hamburg 1, Kennwort: Sammelordner Computer Kurs.

Österreich: Der Sammelordner kostet öS 98. Bitte füllen Sie eine Zahlkarte aus an: Computer Kurs Wollzeile 11, 1011 Wien, Postscheckkonto Wien 7857201 oder legen Sie Ihrer Bestellung einen Verrechnungsscheck bei. Kennwort: Sammelordner Computer Kurs

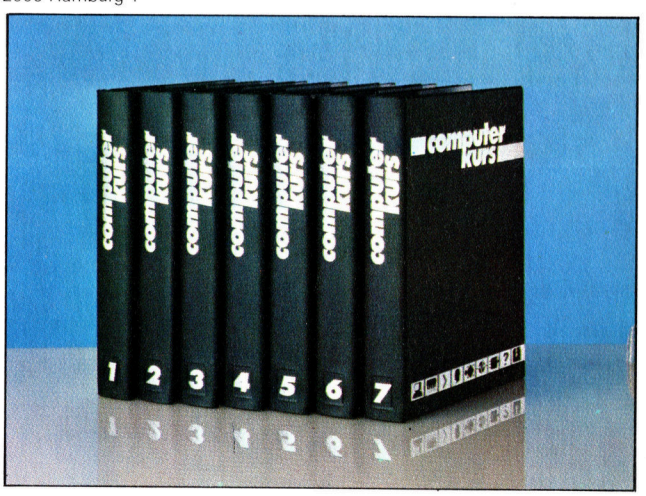
Schweiz: Der Sammelordner kostet sfr 15. Bitte wenden Sie sich an Ihren Kiosk; dort werden Sie jederzeit die gewünschten Exemplare erhalten.

INHALTSVERZEICHNIS

Alle 12 Hefte erscheint ein Teilindex. Die letzte Ausgabe von Computer Kurs enthält den Gesamtindex — darin einbezogen sind Kreuzverweise auf die Artikel, die mit dem gesuchten Stichwort in Verbindung stehen.

Redaktion: Winfried Schmidt (verantw. f. d. Inhalt), Elke Leibinger, Susanne Brandt, Uta Brandl (Layout), Sammelwerk Redaktions-Service GmbH, Paulstraße 3, 2000 Hamburg 1

Vertrieb: Marshall Cavendish International Ltd., Heidenkampsweg 74, 2000 Hamburg 1



Druck: E. Schwend GmbH, Schmollerstraße 31, 7170 Schwäbisch Hall

In den letzten Jahren ist eine ganze Reihe von Thermodruckern auf den Markt gekommen. Einige wenige Modelle wie der Floyd 40 und der Alphacom 32 wurden speziell für den Anschluß an den Sinclair Spectrum entwikkelt. Die meisten anderen (wie der Epson P40 und der Brother HR-5) sind universelle Drukker und benötigen für den Betrieb am Spectrum ein zusätzliches Interface.

Heiße Eisen

Bislang waren Spectrum-Besitzer im wesentlichen auf den ZX-Drucker angewiesen, dessen Lebensdauer und Druckqualität nicht optimal sind. Hier werden Alternativen vorgestellt und eine Reihe wesentlicher Kriterien diskutiert, die vor dem Kauf eines Druckers zu beachten sind.

Eine der ersten Peripherie-Anschaffungen ist der Drucker zur Erstellung von Programmprotokollen. Für Spectrum-Benutzer gibt es bei der Druckerwahl jedoch einige Probleme: Einerseits verfügt der Spectrum über keine der üblichen Druckerschnittstellen, und andererseits mußte man sich beim Sinclair-eigenen ZX-Printer mit einer unsauberen und mit der Zeit ausbleichenden Schrift zufriedengeben. Da Sinclair den Drucker inzwischen nicht mehr produziert, sind andere Hersteller mit preisgünstigen Thermoprintern aus ihrer Fertigung eingesprungen. Einige dieser Geräte sollen hier vorgestellt werden.

Bei der Suche nach einem Drucker sind verschiedene Gesichtspunkte zu berücksichtigen. Im Vordergrund stehen natürlich die Kosten – ein Aspekt, der mehr als nur das Vergleichen von Preisschildern erfordert: Es gibt „verborgene" Zusatzkosten, die mit dem Kaufpreis des Geräts nicht abgedeckt sind. Manche Hersteller behaupten, ihre Drucker seien Spectrum-kompatibel, obwohl das Gerät tatsächlich nur eine RS232C-Schnittstelle aufzuweisen hat. Somit wird stillschweigend vorausgesetzt, daß der Benutzer schon über das Interface 1 verfügt (das den Spectrum mit einer RS232C-Schnittstelle ausstattet) oder es zusätzlich kauft.

Zusätzliche Kosten

Ein anderer verborgener Kostenfaktor ist das Papier. Bei vielen Druckern ist nur Spezialpapier verwendbar – Thermodrucker wie der ZX-Printer zum Beispiel brauchen ein besonderes, temperaturempfindliches Papier, das teurer ist als normales Schreibmaschinenpapier.

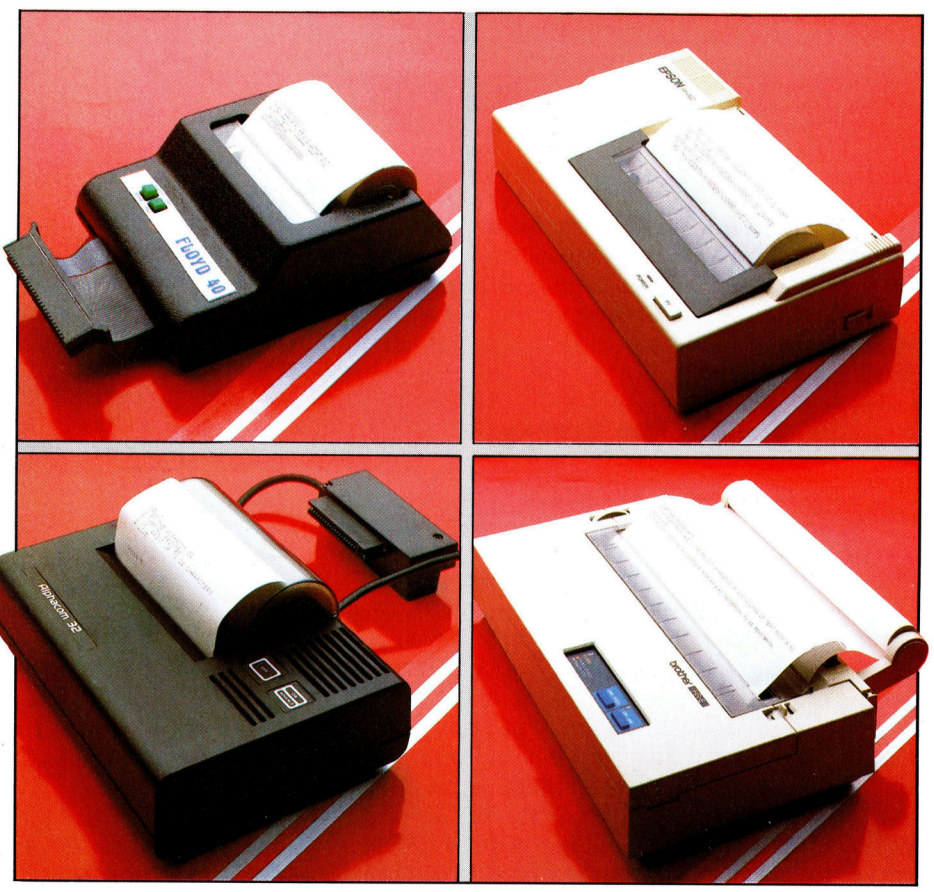
Vor dem Druckerkauf sollten Sie sich daher genau informieren, was das Papier und andere Verbrauchsmaterialien kosten, und ob der Nachschub gesichert ist. Ebenso müssen Sie sich nach dem Service erkundigen – die mechanisch bewegten Teile wie etwa der Druckkopf sind sehr viel störanfälliger als die Elektronik, so daß Sie ein Gerät mit zuverlässigem

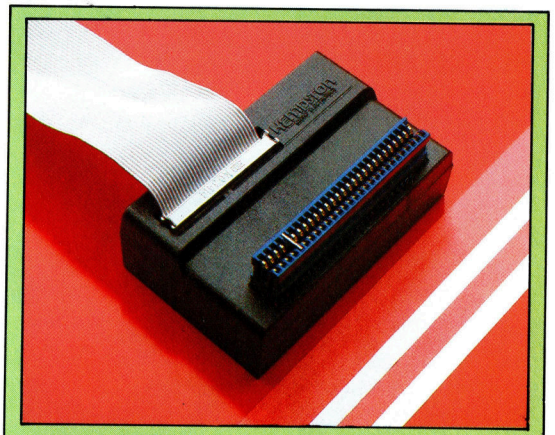
Service bevorzugen sollten.

Schließlich ist noch die Softwarekompatibilität zu beachten. Jeder Drucker ist für bestimmte Steuerzeichen (im allgemeinen im ASCII-Code) programmiert, die für Funktionen wie Wagenrücklauf, Festlegung der Zeilenlänge oder Schriftgröße usw. zuständig sind. Dabei kann es unerfreuliche Überraschungen geben, wenn bei der Anschaffung die Übereinstimmung von Drucker- und Rechnerspezifikationen nicht geprüft wurde.

Sinclair hat, wie die meisten anderen Heimcomputerhersteller, viele der ASCII-Zeichen für spezielle Anwendungen umdefiniert, und ein bestimmtes Steuerzeichen kann trotz eindeutiger Festlegung in der Drucker-Anleitung beim Spectrum eine ganz andere Bedeutung haben. Der einzig sichere Weg besteht darin, sich vor dem Kauf den Druckerbetrieb mit einem Spectrum-Rechner vorführen zu lassen.

Die gleiche Empfehlung gilt für die Software, die Sie benutzen wollen. Nichts wirkt so frustrierend wie Kommandos, die Ihr BASIC anstandslos verarbeitet, die ein bestimmtes Textverarbeitungspaket aber nicht akzeptiert. Unter Umständen muß man den Text zuerst auf Cassette bzw. Diskette abspeichern, das System neu starten und das Geschriebene als sequentielle Datei einlesen, um es anschließend im korrekten Format zu Papier bringen zu können. Speziell bei Texteingaben ist es wichtig, daß der Drucker auch Sonderzeichen und Umlaute darstellen kann.





Kempston-Interface
Für den Betrieb der Drucker, die mit einer Centronics- oder RS232-Schnittstelle ausgestattet sind, muß der Spectrum-Besitzer sich ein geeignetes Interface mit entsprechenden Anschlüssen anschaffen. Die gängigste Schnittstelle ist das Kempston-Interface, das die Verwendung einer Vielzahl Centronics-kompatibler Drucker erlaubt.

Nach diesen allgemeinen Hinweisen nun ein Blick auf einige preisgünstige Thermodrucker für den Sinclair Spectrum: Am preisgünstigsten von den nebenstehend abgebildeten Geräten ist der Alphacom 32. Er erinnert auch am ehesten an den ZX-Printer – die Schrifttype ist die gleiche, ebenso die Zeilenlänge mit 32 Zeichen.

Aus unerfindlichen Gründen hat der Hersteller den Alphacom mit einem separaten Netzteil ausgestattet. Dieses Extrakästchen mit seinen Kabeln ist überflüssig, weil an der Buserweiterung des Spectrum ein 9-Volt-Ausgang zur Verfügung steht, der für jeden Thermodrucker ausreichen sollte.

Alternativen

Der über dem Alphacom abgebildete „Floyd 40" von Shiva Marketing wirkt nicht sehr stabil. Das Gehäuse verbiegt sich ziemlich leicht, und als Halter für die Papierrolle dient nur ein dünner Holzstab. Davon abgesehen ist das Gerät gegenüber dem Alphacom eine deutliche Verbesserung.

Die Spannungsversorgung erfolgt über die Buserweiterung des Spectrum. Außerdem arbeitet der Floyd 40 mit weißem Papier, worauf die Schrift viel besser lesbar ist. Entscheidend hebt sich das Gerät aber dadurch vom Alphacom und vom ZX-Printer ab, daß die Beeinflussung der Ausgabe durch eine Reihe von Steuerzeichen möglich ist.

Diese Zeichen werden an den Drucker mit einem normalen LPRINT-Befehl übergeben. Zusätzlich wird das Steuerzeichen zwischen Ausrufezeichen gesetzt, damit es nicht gedruckt, sondern als Befehl interpretiert wird. Der Befehl LPRINT"!H!" veranlaßt eine Umschaltung auf doppelte Schrifthöhe; Wiederholung dieses Befehls bewirkt Rücksetzen.

Ferner sind die Wiedergabe von Grafiksym-

bolen und die Ausgabe in doppeltbreitem oder invertiertem Format vorgesehen, außerdem ein Grafikmode, in dem der 5 × 7-Punkt-Druckkopf Bilder erzeugen kann. Das System arbeitet ziemlich langsam, aber das Schriftbild ist ebenso gut wie bei vielen Geräten, die das Doppelte kosten. Einziger echter Nachteil ist die Beschränkung auf eine Papierbreite von 80 mm.

Das dritte abgebildete Gerät ist der Epson P40, den es als P40S (Serielle Version) mit RS232C-Buchse und als P40P mit Centronics-Parallel-Port gibt. Da der Sinclair Spectrum weder das eine noch das andere hat, ist die Anschaffung des Interface 1 oder eine der vielen anderen auf dem Markt befindlichen Schnittstellen zwingend.

Schnittstellen-Probleme

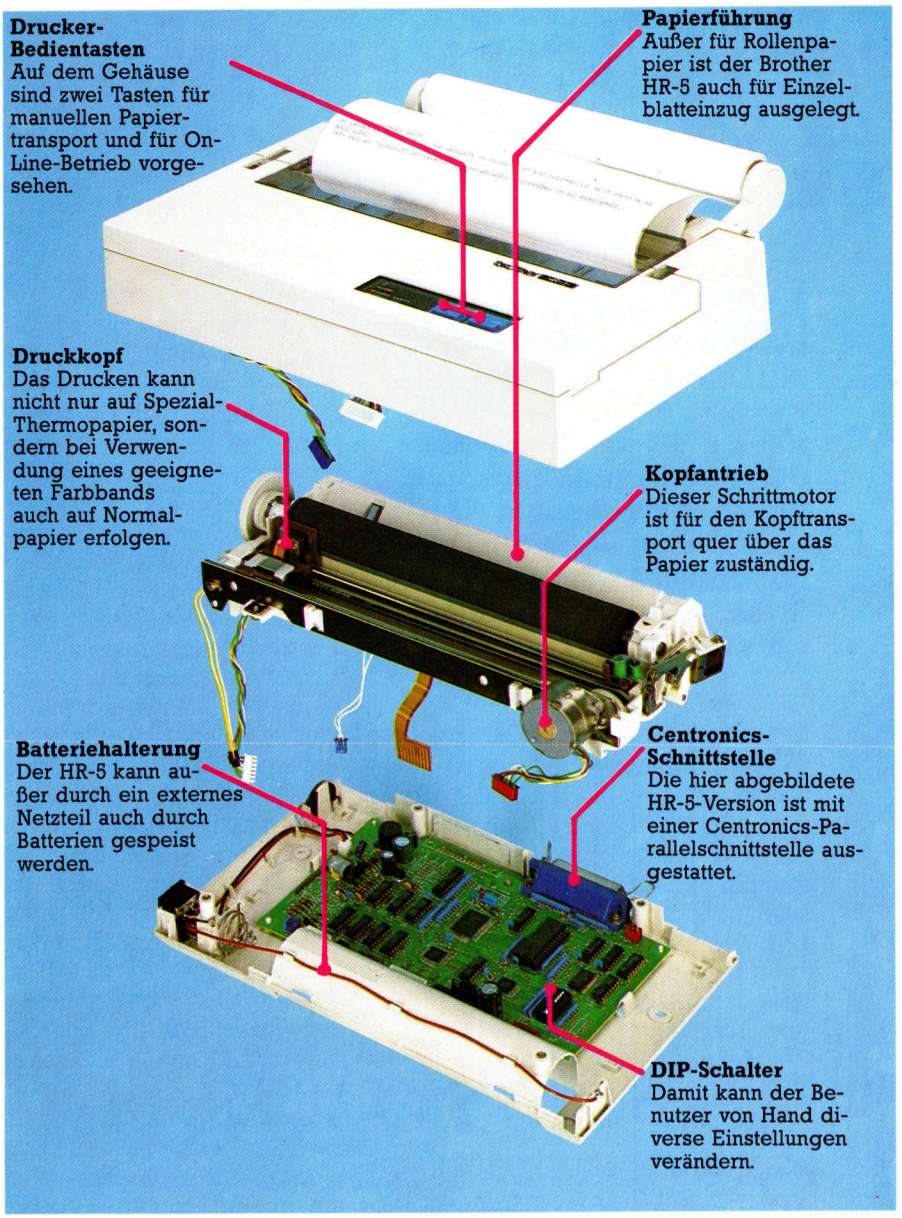
Ein Problem bereitet beim Interface 1 die RS232C-Buchse, weil sie für einen nicht normgerechten siebenpoligen D-Stecker ausgelegt ist. Aus der Interface- und der Drucker-Anleitung geht aber die richtige Stiftbelegung hervor, und der Rest ist reine Lötarbeit. Wenn Sie davor aber zurückscheuen, sollten Sie sich besser eine der angebotenen Schnittstellen kaufen. Von Kempston beispielsweise gibt es ein Interface mit passendem Stecker für den Spectrum und Normanschlüssen für RS232- und Centronics-Geräte.

Trotz seiner relativ bescheidenen Abmessungen kann der P40 wahlweise 40 oder 80 Zeichen pro Zeile drucken. Er akzeptiert auch eine große Anzahl von Escape-Codes, die sonst nur von größeren Druckern verarbeitet werden. Allerdings hat der Spectrum keine spezielle Escape-Taste, so daß diese Befehle als ASCII-Zeichen in der Form CHR$(27);"E"; übergeben werden müssen. Daraufhin schaltet der P40 entsprechend um – CHR$(27) stellt das ASCII-Escape-Symbol dar.

Beim P40 können Sie außerdem den Zeichensatz variieren und einen „Bit Image Mode" (freie punktweise Zeichengestaltung) sowie unter anderem komprimierte Schrift wählen. Abgesehen von diesen software-gesteuerten Optionen sind außerdem noch einige DIP-Schalter für die Vorgabe von Parität und Zeilenlänge vorgesehen.

Anders als die bisher besprochenen Geräte arbeitet der HR-5 von Brother wahlweise mit Thermopapier oder (bei Verwendung eines Farbbands) mit Normalpapier. Außerdem reicht die Wagenbreite bei diesem Drucker für A4-Format aus, so daß man Briefe und andere gängige Textverarbeitungsaufgaben damit erledigen kann; die Zeilenlänge kann bis zu 132 Zeichen betragen. Das Gerät arbeitet dabei fast lautlos.

Wie der P40 von Epson macht auch der Brother für die Druckformatierung reichlich von Escape-Codes Gebrauch; er kann so außerdem sehr viele Sonderzeichen (auch fremdsprachliche) wiedergeben. Eine andere Gemeinsamkeit des HR-5 und des P40 besteht in der Schnittstellenausstattung – wahlweise Centronics oder RS232C, so daß für den Betrieb am Spectrum leider wieder ein zusätzliches Interface anzuschaffen ist.



Beim Druckerkauf gibt es eigentlich nur Probleme, wenn Sie sich für ein Gerät entscheiden, das mehr kann als die Spezialanfertigungen für den Spectrum. Der Alphacom und der Floyd 40 haben den Vorteil, daß sie direkt angeschlossen werden können und die üblichen Spectrum-Kommandos wie COPY akzeptieren; sie sind aber eigentlich nur für den Ausdruck von Listings zu gebrauchen.

Die teureren Drucker mit mehr Möglichkeiten sind für den Anschluß an die verschiedensten Rechnermodelle gedacht und nicht nur für den Sinclair Spectrum. Daher bleibt Ihnen die Mühe nicht erspart, derartige Geräte an die Spectrum-Erfordernisse anzupassen.

PROgramming in LOGic

Die Programmiersprache PROLOG beruht auf der einfachen Aussagenlogik, mit der wir auch die meisten Probleme des täglichen Lebens lösen. Die Sprache ist praktisch und leicht überschaubar. Unsere Prolog-Serie beginnt mit einem Einblick in die Struktur der hier angewandten Logik.

Es scheint schon lange her zu sein, daß die Japaner verkündeten, sie würden mit der Entwicklung der „Fünften Computergeneration" die Microtechnologie von Grund auf verändern. Damals wurde eine weitreichende Entscheidung getroffen: Die wenig bekannte Programmiersprache PROLOG sollte das „zentrale Sprachsystem" für die geplanten schnellen und intelligenten Datenbankmaschinen werden.

PROLOG ist ein Akronym für „**Pro**gramming in **Log**ic". Die Sprache ist eine gute, wenn auch nicht perfekte Verwirklichung dieses Ideals. Es stellt sich jedoch die Frage: Warum überhaupt mit direkter Logik programmieren? Es gibt viele Logiksysteme, die sich für die Beschreibung der Welt und aller ihrer Eigenschaften einsetzen lassen. Einige sind uns vertraut – beispielsweise die Mathematik – andere sehen sehr exotisch aus, wie etliche philosophische Lehren. Die einfache Aussagenlogik der Integralrechnung ist unserem täglichen Denken sehr ähnlich, obwohl sie mit einem eigenen Zeichensystem arbeitet und etlichen Einschränkungen unterworfen ist. Eine Aussage läßt sich am leichtesten als die Beziehung zwischen Dingen ansehen. In dem Satz „Peter mag Anna" ist die Satzaussage das Wort „mag". Wir könnten diese Aussage auch als „mag(Peter, Anna)" schreiben. Dies ist zwar weniger gut lesbar, macht aber deutlich, was Aussage und was Gegenstand ist. Die Tatsache, daß Peter männlich ist, ließe sich „männlich(Peter)" schreiben, wobei „männlich" eine Aussage ist, die den Gegenstand (Peter) betrifft.

Zwar sind dies nur einfache Darstellungen von Tatsachen, doch zeigt eine Erweiterung der Logik, wie einige Tatsachen andere Inhalte mit einschließen. Mit der Aussagenlogik können wir die Welt als „Tatsachen" und „Folgerungen" beschreiben und aus diesen Beschreibungen wieder neue Tatsachen folgern. Dazu benötigen wir Variablen. Logische Variablen sind den vertrauten BASIC-Variablen ähnlich, ihr Gültigkeitsbereich beschränkt sich jedoch auf den Satz (die Tatsache oder Folgerung), in dem sie auftauchen, und gilt nicht für alle Sätze. Der Peter, der Anna mag, kann also ein anderer sein als der, der männlich ist.

Eine Tatsache läßt sich als „weiblich(X)→ mag (Peter,X)" schreiben. Der Pfeil bedeutet „bedingt", so daß der Satz eine Regel enthält, die aussagt „Die Tatsache, daß X weiblich ist, bedingt, daß Peter X mag". In normalem Deutsch würde das bedeuten, Peter mag X, wenn X weiblich ist. Diese Regel ist ein Beispiel für den „Hornsatz", eine bestimmte Satzart der Aussagenlogik der Integralrechnung. Hornsätze haben als Kopf eine Aussage (die Konsequenz), die nur dann wahr ist, wenn alle Aussagen des Hauptteils (die Vorbedingungen) wahr sind.

A wenn B und C und D
ist ein Hornsatz mit dem Kopf A und dem Hauptteil B, C und D. Eine einfache Tatsache läßt sich als Konsequenz ohne Vorbedingungen verstehen und wird als wahr angesehen.

Unter der Aussagenlogik werden Programme zu Sammlungen von Tatsachen und Regeln, die Dinge beschreiben, an denen wir interessiert sind. Die Form dieser Beschreibungen ähnelt unserer Wahrnehmung des Problems. Logikprogramme versuchen nun, die Wahrheit oder Unwahrheit von Aussagen zu beweisen. Ist die Aussage eine einfache Tatsache, können wir ohne weiteren Aufwand vermuten, daß sie wahr ist. Ist sie aber die Konsequenz einiger Regeln, müssen wir erst die Wahrheit aller Vorbedingungen beweisen, bevor wir wissen, ob die Aussage wahr ist. Wenn wir daher feststellen möchten, ob Peter männlich ist, müssen wir die Aussage „männlich(Peter)" beweisen. Wollen wir wissen, ob Anna Peter mag, müssen wir zuerst die Aussage „männlich(Peter)" und dann „mag(Peter, Anna)" beweisen.

Colmerauer und Kowalski

PROLOG wurde in den frühen siebziger Jahren an der Marseiller Universität von A. Colmerauer entwickelt und basiert zum Teil auf der Arbeit von Bob Kowalski, der jetzt am Imperial College in London ist. Die Sprache verwendet nur den Hornsatz der Aussagenlogik. Ihre Schreibweise entspricht unseren Beispielen. Es wäre möglich gewesen, die gesamte Aussagenlogik der Integralrechnung darin zu verwirklichen, doch PROLOG ist – wie jede andere Programmiersprache –

PROLOG-Dialekte

Da PROLOG viel RAM benötigt, gibt es nur wenige Versionen für Heimcomputer. Für den Spectrum steht jedoch MICRO-PROLOG von Logic Programming Associates zur Verfügung.

Die Besitzer anderer Micros müssen sich damit trösten, daß PROLOG mehr und mehr Interesse unter den Softwareherstellern findet und Versionen zum Beispiel für den Apple, Acorn B und den Enterprise in Vorbereitung sind.

Das MICRO-PROLOG des Spectrum unterscheidet sich in mancher Hinsicht vom Standard-DEC-10-PROLOG – die Version, die wir für die Beispiele unserer Serie verwenden. Wir werden jedoch eine Reihe von kurzen PROLOG-Varianten abdrucken, so daß die Programme auch auf dem MICRO-PROLOG des Spectrum laufen. Die wichtigsten Unterschiede erklären wir in einer späteren Folge.

ein Kompromiß zwischen Computerumgebung und den Fähigkeiten des Sprachsystems. Zwar wird im Augenblick noch erforscht, wie dieses Gleichgewicht verbessert werden kann, doch scheint die augenblickliche Struktur von PROLOG recht dauerhaft zu sein.

DEC-10 PROLOG kommt einer Standardversion am nächsten. Sie erhielt diesen Namen, da die Sprache zu Beginn auf einem Großcomputer der Firma Digital Equipment Corporation lief. Die „Anwenderbibel" für PROLOG ist ein Buch von Clocksin und Mellish mit dem Titel „Programming in PROLOG", das den Sprachstandard mit vielen praktischen Einzelheiten beschreibt. Fast alle laufenden Versionen von PROLOG, darunter auch C-PROLOG, sind diesem Standard nachempfunden, obwohl es viele Varianten und Dialekte gibt. Für Microcomputer gibt es zwei Systeme: Die Version von Expert Systems kommt dem Standard sehr nahe, während MICRO PROLOG von Logic Programming Associates zwar beliebt ist, in seiner Syntax und inneren Struktur aber bedeutend vom Standard abweicht. PROLOG benötigt viel Speicherplatz und läßt sich daher nur schwer in Micros mit weniger als 64KByte RAM unterbringen.

Das Backtracking

PROLOG-Programme funktionieren nicht nach dem vertrauten Ablauf, in dem zuerst der erste Befehl ausgeführt wird, dann der zweite etc. bis zum letzten Befehl, sondern mit einer Technik, die „Backtracking" genannt wird.

Bei der Lösung eines Problems arbeitet sich PROLOG durch eine Kette von Regeln hindurch und setzt sich bei jedem Schritt ein neues Ziel, das zu erfüllen ist. Erweist sich in der Kette ein bestimmter Weg als Sackgasse, geht PROLOG auf eine frühere „Wegkreuzung" zurück und verzweigt in eine andere Richtung.

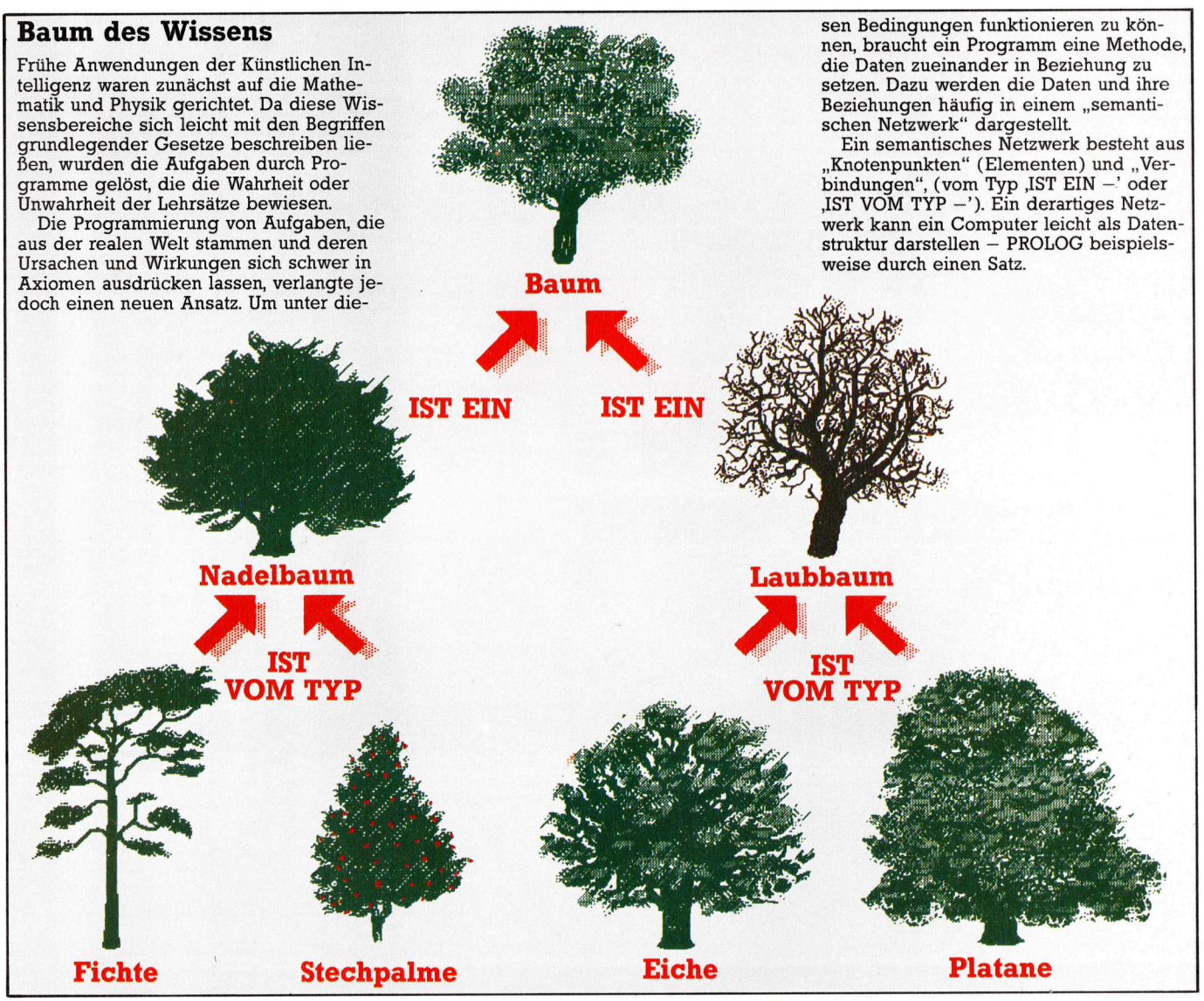
Diese Vorgehensweise gibt PROLOG ein „Gesicht", das sich von dem anderer Programmiersprachen wesentlich unterscheidet. Aufgrund der deklarativen Natur von PROLOG werden die Regeln wie Sätze der Aussagenlogik der Integralrechnung gelesen. So ist beispielsweise X ein Onkel von Y, wenn X männlich ist, X ein Abkömmling von Z und Z ein Elternteil von Y. In PROLOG läßt sich diese Aussage auch auf vertrautere Weise ausdrücken: Um zu beweisen, daß X ein Onkel von Y ist, muß gezeigt werden, daß X von Z abstammt und Z ein Elternteil von Y ist. Diese deklarative Schreibweise, die in fast allen anderen Programmiersprachen fast völlig fehlt, ist ein wichtiges Hilfsmittel für das Verständnis des Programmaufbaus.

Baum des Wissens

Frühe Anwendungen der Künstlichen Intelligenz waren zunächst auf die Mathematik und Physik gerichtet. Da diese Wissensbereiche sich leicht mit den Begriffen grundlegender Gesetze beschreiben ließen, wurden die Aufgaben durch Programme gelöst, die die Wahrheit oder Unwahrheit der Lehrsätze bewiesen.

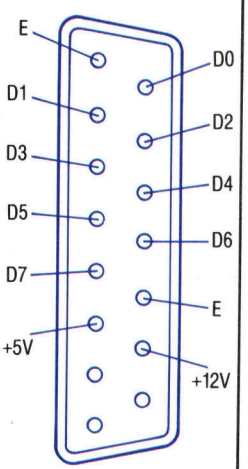
Die Programmierung von Aufgaben, die aus der realen Welt stammen und deren Ursachen und Wirkungen sich schwer in Axiomen ausdrücken lassen, verlangte jedoch einen neuen Ansatz. Um unter diesen Bedingungen funktionieren zu können, braucht ein Programm eine Methode, die Daten zueinander in Beziehung zu setzen. Dazu werden die Daten und ihre Beziehungen häufig in einem „semantischen Netzwerk" dargestellt.

Ein semantisches Netzwerk besteht aus „Knotenpunkten" (Elementen) und „Verbindungen", (vom Typ ‚IST EIN –' oder ‚IST VOM TYP –'). Ein derartiges Netzwerk kann ein Computer leicht als Datenstruktur darstellen – PROLOG beispielsweise durch einen Satz.



Montage

Nach dem Zusammenbau des Antriebs und dem Bestücken der dazugehörigen Platine können wir mit der Verdrahtung beginnen: Die Motoren werden an die Platine angeschlossen. Über einen D-Stecker am Gehäuse des Roboters und ein einfaches Interface wird die Verbindung zum User Port des Computers hergestellt.



D-Buchse, von hinten gesehen

Steckermontage

Die Skizze zeigt die richtige Verbindung von D-Stecker und User-Port-Anschlüssen. Für den Acorn B wird 20-adriges Flachkabel mit einem 20-poligen IDC-Stecker (Schneidstekker) verwendet. Auf der anderen Seite des Kabels werden die benutzten (elf) Adern abisoliert, verzinnt und mit der Interface-Platine verlötet.

Wer mit dem C 64 arbeitet, braucht einen 24-poligen Platinenstecker und ein Stück 12-adriges Flachkabel. Die einzelnen Adern am Stecker müssen exakt mit der Bezeichnung der Leitungen an der Interface-Platine übereinstimmen. Markieren Sie die Oberseite des Steckers, damit er nicht versehentlich falsch herum angeschlossen wird.

Alles geschafft? Dann kann der „Roboter" über das Interface mit dem User Port verbunden werden. Die 12-Volt-Stromversorgung wird an der 2,1 mm-Buchse auf der Interface-Platine angeschlossen.

Bit für Bit

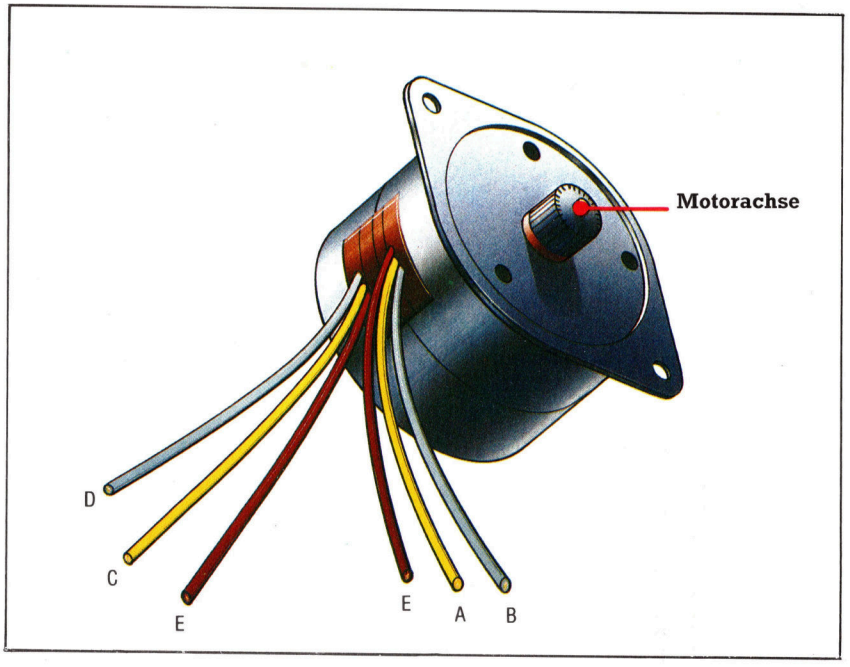
Fertig? Ein kleines Programm bringt Ihrem Robot-Auto das Laufen bei: Die Bits 0 bis 3 des User-Port-Datenregisters steuern die Motoren. Bit 0 ist das RESET-Bit, das normalerweise auf 1 liegt. Bit 1 und 2 regeln die Laufrichtung des rechten und linken Motors und Bit 3 dient als Impuls-Bit, das die Motoren um je einen Schritt weiterdrehen läßt. Zum Anwählen der Richtung dienen die Tasten T, B, F und H. Das Impuls- oder Triggerbit wird durch eine Programmschleife erzeugt.

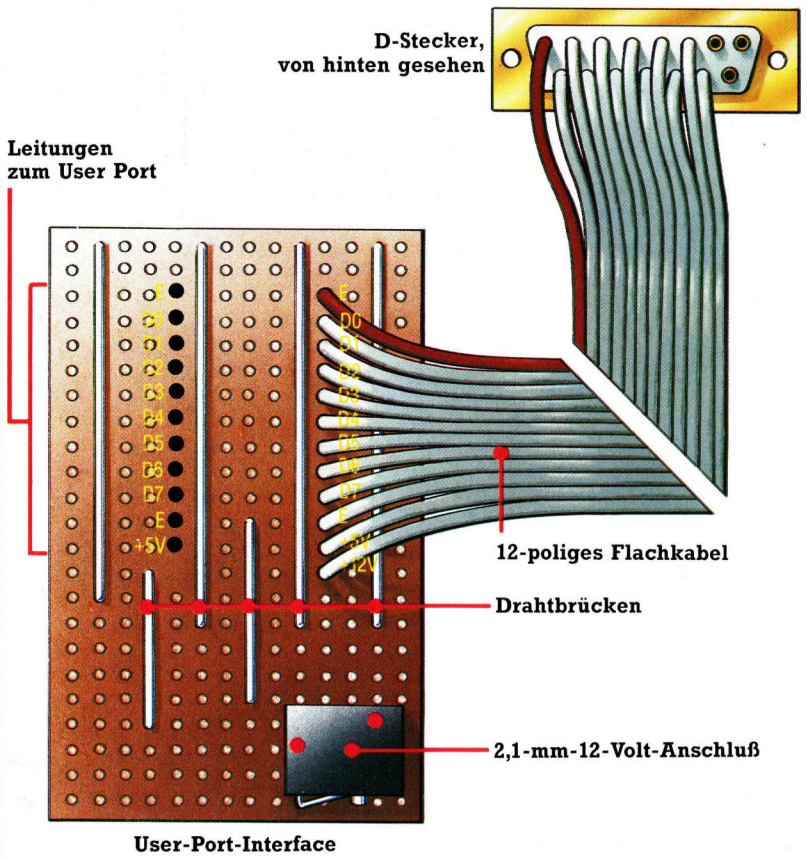
```
1000 REM **** BBC ROBOT CONTROLLER ****
1010 DDR=&FE62:DATREG=&FE60:?DDR=15:REM LINES 0-3 OUTPUT
1020 PROCinitialise:REPEAT
1030 A$=INKEY$(1):IF A$<>"" THEN PROCtest_keyboard
1040 PROCpulse(10)
1050 UNTIL A$="X":?DATREG=0:END
1060 DEF PROCinitialise
1070 forwards=4:backwards=2:left=6:right=8
1080 dir=forwards:?DATREG=dir+1:ENDPROC
1100 DEF PROCpulse(m)
1110 FOR c=1 TO m
1120 ?DATREG=(?DATREG OR 8):PROCdelay(2)
1130 ?DATREG=(?DATREG AND 247):PROCdelay(2)
1140 NEXT c:ENDPROC
1150 DEF PROCdelay(n)
1160 FOR I=1 TO n:NEXT I
1170 ENDPROC
1180 DEF PROCtest_keyboard
1190 IF A$="T" THEN dir=forwards
1200 IF A$="B" THEN dir=backwards
1210 IF A$="F" THEN dir=left
1220 IF A$="H" THEN dir=right
1230 ?DATREG=((?DATREG AND 249)OR dir)
1240 ENDPROC


10   REM **** CBM 64 ROBOT CONTROLLER ****
20   DDR=56579:DATREG=56577:POKEDDR,15
30   GOSUB1000:REM INITIALISE
40   GETA$:IFA$<>"" THEN GOSUB3000:REM KEYS
50   M=10:GOSUB1500:REM PULSE
60   IFA$<>"X" THEN 40
70   POKEDATREG,0:END
1000 REM **** INITIALISE S/R ****
1010 FW=4:BW=2:LF=6:RT=8
1020 DR=FW:POKEDATREG,DR+1:RETURN
1500 REM **** PULSE S/R ****
1510 FOR C=1 TO M
1520 POKEDATREG,(PEEK(DATREG)OR8):GOSUB2000:REM DELAY
1530 POKEDATREG,(PEEK(DATREG)AND247):GOSUB2000:REM DELAY
1540 NEXT C:RETURN
2000 REM ****DELAY S/R ****
2010 FOR I=1 TO N:NEXT I:RETURN
3000 REM **** KEYBOARD TEST S/R ****
3010 IF A$="T" THEN DR=FW
3020 IF A$="B" THEN DR=BW
3030 IF A$="F" THEN DR=LF
3040 IF A$="H" THEN DR=RT
3050 POKEDATREG,((PEEK(DATREG)AND249)ORDR)
3060 RETURN
```

Was noch fehlt . . .

Zum korrekten Anschluß der Motoren und des D-Steckers benötigen Sie die Abbildung der Platine (Seite 1253). Sie zeigt, wie die Leitungen richtig an der Platine festgelötet werden. Zuerst die Motoren: Aus dem Gehäuse kommen insgesamt sechs Drähtchen, von denen jeweils drei und drei zusammengehören.

In beiden Gruppen gibt es zwei zusammengehörige Leitungen (gelb und grau) und eine rote Leitung. Die mit „A" und „B" bezeichneten gelben und grauen Leitungen kommen auf der zur Motorachse gelegenen Seite des Gehäuses heraus (siehe Zeichnung). Beachten Sie beim Anlöten der Anschlüsse auf der Platine genau die Buchstabenkennung!





User-Port-Interface

Liste der Bauteile

Anzahl	Bauteil
1	15-polige D-Buchse
1	Gehäuse für D-Buchse
1	2,1 mm-Buchse (12-Volt-Anschluß)
1	20-poliger IDC-Stecker (Acorn B)
1	24-poliger Platinenstecker (C 64)
1	Rolle Klebeband
4 Meter	12-poliges Flachkabel
1 Meter	20-poliges Flachkabel (Acorn B)
1	Gleichspannungs-Netzteil 12 V, 1A

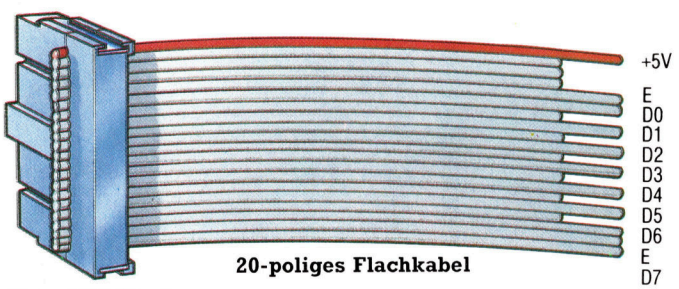
User-Port-Anschluß

Wenn die Verdrahtung des Roboters fertiggestellt ist, brauchen wir für den Anschluß am User Port eine einfache Interface-Platine. Zur Versorgung der Schrittmotoren dient eine 12-Volt-Gleichspannungsquelle.

An einem Stück Lochplatine (Veroboard, 24 Streifen à 14 Löcher) wird 3 Meter langes, 12-poliges Flachkabel angelötet. Die rote Kabelader dient dem Anschluß an die D-Buchse als Orientierung. Das Buchsengehäuse brauchen wir zum Schutz der Lötstellen und als Zugentlastung für das Flachkabel.

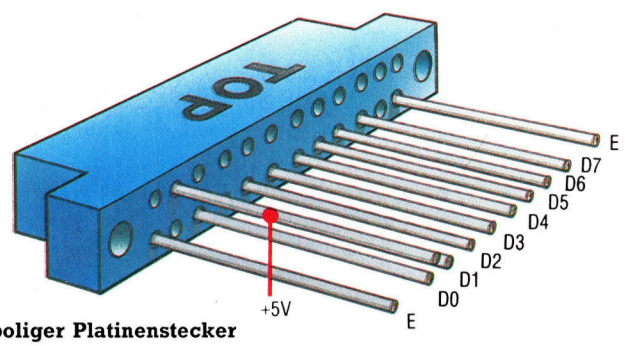
Die Buchse für das Netzteil wird auf die Platine gelötet. (Polarität beachten!) Der innenliegende Anschluß der Buchse ist Minus. Drahtbrücken nach der Zeichnung einsetzen. Links unten sind die User-Port-Anschlüsse für den Acorn B und den C 64 abgebildet.

Acorn B



20-poliger IDC-Stecker

Commodore 64



24-poliger Platinenstecker

Verdrahten und Verkabeln

Nach dem Anschluß der beiden Motoren ist noch die Verbindung zum D-Stecker auf der Oberseite des Robotergehäuses herzustellen. Auf der Skizze sind die Anschlußpunkte so eingezeichnet, wie man sie von der Unterseite des Gehäusedeckels her sieht. Ein 12-poliges Flachkabel führt vom Stecker mit den Datenleitungen D0 bis D3, +5 Volt, +12 Volt und Masse zur Platine. Auch hier brauchen Sie die Darstellung der Platine von Seite 1253, auf der die richtigen Lötpunkte eingezeichnet sind. Achten Sie besonders auf die 12-Volt-Leitung – bei einem Fehlanschluß könnte Ihr Computer beschädigt werden! Die Datenleitungen D4 bis D7 werden jetzt noch nicht verwendet, sie dienen beim weiteren Ausbau des Roboters zum Anschluß von Sensoren.

Die Verdrahtung im Roboter ist jetzt vollständig. Suchen Sie im Gehäuse einen geeigneten Platz für die Platine und befestigen Sie diese mit Klebestreifen. Danach kann der Deckel aufgesetzt und mit den beigelegten Schrauben fixiert werden.

Zahlenspiele

Spielziel dieses kurzen BASIC-Programms ist, eine Reihe von Zahlen mit möglichst wenigen Zügen in aufsteigende Reihenfolge zu bringen. Dieser Algorithmus läßt sich anschließend beliebig abwandeln.

Reverse

```
 20 DIM a(20)
 30 CLS : PRINT "Reverse !"
 40 INPUT "How many numbers ? ";n
 50 IF n<0 OR n>20 OR n<>INT n THEN  GO TO 30
 60 REM Jumble the list
 70 FOR i=1 TO n: LET a(i)=i: NEXT i
 80 RANDOMIZE
 90 FOR i=1 TO n
100 LET r=INT (RND*n+1)
110 LET x=a(r): LET a(r)=a(i): LET a(i)=x
120 NEXT i
130 LET t=1
135 REM Print the board
140 CLS : PRINT "Turn ";t;": The list is:": PRINT
150 FOR i=1 TO n: PRINT a(i);" ";: NEXT i
152 REM Check for a win
154 LET i=1
156 IF a(i)=i THEN  LET i=i+1: IF i<=n THEN  GO
TO 156
158 IF i>n THEN  GO TO 230
159 REM Get a go
160 PRINT : PRINT : INPUT "Reverse?";r
170 IF r<>INT r OR r<0 OR r>n THEN  GO TO 140
175 REM Reverse r
180 LET t=t+1
190 FOR i=1 TO INT (r/2)
200 LET x=a(i): LET a(i)=a(r-i+1): LET a(r-i+1)=x
210 NEXT i
220 GO TO 140
230 REM A Winner!
240 PRINT : PRINT : PRINT "You finished in ";t;"
turns"
250 PRINT : INPUT "Play again (y/n) ? ";a$
260 IF a$="Y" OR a$="y" THEN  RUN
270 CLS : STOP
```

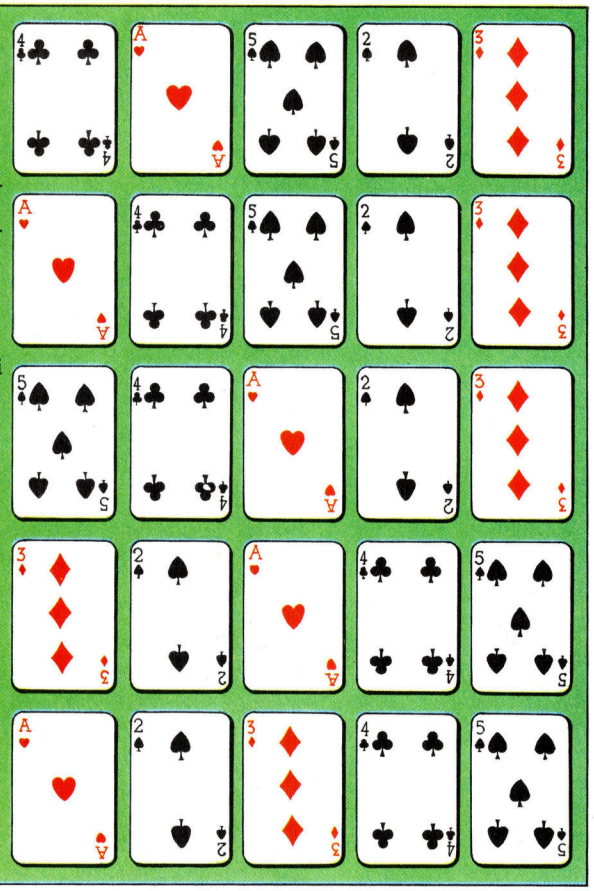
BASIC-Dialekte

Ersetzen Sie beim Commodore 64 und VC 20 RANDOMIZE durch XX=RND(-TI). Ersetzen Sie RND*N durch RND(1)*N sowie CLS durch PRINT CHR$(147).

Beim Acorn B löschen Sie Zeile 80 und ersetzen R-INT(RND*N+1) durch R=RND(N).

Das Spiel

Das Ziel des Spiels ist es, eine Reihe von Objekten zu sortieren, indem Untergruppen umgekehrt werden. Die umzukehrende Untergruppe muß immer mit dem ganz links liegenden Element beginnen. Die richtige Sequenz für unser Beispiel ist 2-3-5-3. Das bedeutet: Kehre die links befindlichen zwei Karten um, dann die links liegenden drei Karten usw.



Das Programm generiert eine Liste mit Zufallszahlen. Die richtige Reihenfolge der Zahlen kann nur durch Umkehren von Zahlengruppen innerhalb der Liste erreicht werden. Nehmen wir an, der Computer generiert auf Angabe des Spielers die folgende Liste mit neun zufälligen Zahlen:

2 8 4 7 1 5 6 9 3

Gibt der Spieler „REVERSE? 5" an, werden die ersten fünf Zahlen umgedreht, worauf die Liste wie folgt aussieht:

1 7 4 8 2 5 6 9 3

Sie werden jetzt sicher vermuten, daß solche Gedankenspiele mit einem Algorithmus leichter zu lösen sind. Das ist richtig, doch es ist sehr schwer, einen guten Algorithmus zu entwickeln. Nehmen wir an, es befinden sich n Zahlen in der Liste. Der naheliegendste Algorithmus wäre dann:

- Finde die größte Zahl der Liste und kehre alle Zahlen bis zu ihrer Position um. Die größte Zahl befindet sich nun auf der linken Seite.
- Kehre alle Zahlen um, so daß sich die größte Zahl an der Position auf der rechten Seite befindet. Dies erfordert einen zweiten Zug.
- Finde die zweitgrößte Zahl und wiederhole den eben beschriebenen Vorgang. Um diese Zahl an die gewünschte Position zu bringen, ist eine Umkehrung von n-1 erforderlich.
- Wiederhole den Vorgang so oft, bis die Reihenfolge erreicht ist.

Dieser Algorithmus löst das Puzzle immer in 2n-3 Zügen. Doch ist eine Lösung auch mit weniger Zügen möglich. Betrachten Sie hierzu unser Beispiel. Der Algorithmus würde sieben Züge benötigen (2 × 5-3), ein geübter Spieler lediglich vier.

Das hier gezeigte Programm ist nur ein Beispiel von einer ganzen Reihe von Umkehrspielen. Vielleicht wollen Sie auch ein Spiel entwickeln, bei dem zum Beispiel ein Raster anstelle einer Zeile in eine bestimmte Ordnung gebracht werden soll. Eventuell können Sie verschiedene farbige Blöcke anstelle der Zahlen verwenden. Ziel des Spieles wäre dann, ein bestimmtes farbiges Muster zusammenzustellen. Außerdem können Sie einen Algorithmus in Ihr Programm integrieren, der einem Spieler, der Schwierigkeiten bei der Lösung der Aufgabe hat, Hilfestellungen geben kann.

Schulkamerad

Research Machines ist als Hersteller des Link 380Z bekannt. Dieser Rechner wurde ursprünglich für Forschungs- und Entwicklungszwecke konstruiert und gelangte dann an englischen Schulen zu außerordentlicher Beliebtheit. Danach brachte das Unternehmen einen weiteren Microcomputer heraus, den Link 480Z.

Der Link 480Z ist als Netzwerk- oder Einzelgerät erhältlich. Hier beschäftigen wir uns mit der Standardausführung. Auf den ersten Blick unterscheidet sich die Maschine stark vom Vorgängermodell. Der 380Z bestand aus einem großen schwarzen Metallgehäuse, in dem sich der Rechner und die Diskettenstationen befanden. Dieses wurde durch ein Kabel mit dem externen Keyboard verbunden. Der Link 480Z dagegen ist mit einem stabilen Kunststoffgehäuse ausgestattet, in das die Tastatur integriert ist. Und die Diskettenstation wird als externe Einheit angeboten.

Die Maschine ist mit einer Standard-QWERTY-Tastatur ausgestattet. Die Tasten machen einen verläßlichen und sicheren Eindruck. Die Steuerungstasten, einschließlich der Zeilentaste und der Repeat-Taste für den Bildschirm bzw. für Editier-Funktionen, befinden sich links und rechts außen neben der Tastatur. Rechts außen befindet sich das Cursor-Steuerungsfeld mit den Funktionstasten.

Dank der zahlreichen Schnittstellen auf der Rückseite des Rechners kann der Computer mit vielen Peripheriegeräten verbunden werden. Ganz links liegt der RF-Anschluß, mit dem der Rechner an einen Fernseher angeschlossen werden kann. Rechts außen befindet sich der Reset-Schalter.

Der Link 480Z ist mit zwei verschiedenen Monitoranschlüssen ausgestattet: Mit der fünfpoligen DIN-Buchse kann der Computer an einen herkömmlichen Monitor angeschlossen werden. Darüber befindet sich eine weitere DIN-Buchse für andere TTL- und RGB-Monitore. Über eine zusätzliche serielle Schnittstelle können weitere externe Geräte angeschlossen werden.

Rechts neben dem Cassettenrecorderanschluß liegt das parallele Input/Output-Inferface. Diese Schnittstelle entspricht zwar nicht dem Centronics-Standard, ist aber Centronics-kompatibel. Die Pins liegen jedoch nicht in der Centronics-üblichen Reihenfolge. Durch Umbelegung kann aber ein Standard-Centronics-Anschluß hergestellt werden.

Darüber hinaus verfügt der Link 480Z über zwei RS232-Schnittstellen, womit der Rechner sowohl mit seriellen Druckern als auch mit einer Doppeldiskettenstation verbunden werden kann. Neben den seriellen Schnittstellen liegen zehn DIP-Schalter. Der erste, mit „R" gekennzeichnet, gibt die Möglichkeit, die RESET-Taste zu deaktivieren. Mit dem zweiten wird der interne Lautsprecher – unter der Tastatur angebracht – ein- oder ausgeschaltet.

Mit den acht an der äußeren linken Seite des Rechners befindlichen DIP-Schaltern kann der Anwender Netzwerk-Adressen einstellen. Sie werden als Binär-Zahl gelesen und identifizieren den Computer, wenn er in einem Netzwerk verkabelt ist. Da der 480Z über acht dieser Schalter verfügt, können bis zu 256 Geräte mit ihm vernetzt werden. Das Netzwerk-Kabel selbst ist an einer Videobuchse auf der Rückseite des Computers angebracht.

Intelligente Floppy

Die MD2-Doppeldiskettenstation ist mit dem Computer über ein serielles statt, wie sonst üblich, einem parallelen Interface verbunden. Sie wird am zweiten RS232-Port angeschlossen. Die Übertragungsrate beträgt dennoch 38,5 KBaud und ist somit der bei Micros mit parallelem Datentransfer vergleichbar. Die Doppeldiskettenstation verwendet die üblichen 5¼-Zoll-Disketten (doppelte Dichte und doppelseitig). Das Diskettenstationsgehäuse ist aus

Die Tastatur und das Kunststoffgehäuse vermitteln ein eleganteres Bild als das Vorläufermodell, der 380Z, wenngleich der Rechner nach modernen Maßstäben noch immer eine große Maschine ist. Das liegt an den beiden Platinenlagen im Rechner. Eine dient nur für die Hauptfunktionen, die andere für die Netzwerk-Technik.

demselben soliden Kunststoffmaterial gefertigt wie das des Computers.

Auf der Rückseite der Diskettenstation befindet sich ein Paar RS232-Buchsen. Über die eine erfolgt die Verbindung zum 480Z, die andere ermöglicht die „Verkettung" der verschiedenen Peripheriegeräte miteinander. Die Station verfügt außerdem über eine eigene Stromversorgung. Das Diskettenverwaltungs-System, mit dem die Datenübertragung vom und zum Computer erfolgt, befindet sich im Laufwerk. Beim Einsatz dieser „intelligenten" Diskettenstation kann der Rechner nebenbei andere Arbeiten erledigen, da die Steuerung durch die Diskettenstation selbst erfolgt. Somit steht mehr freier Speicherplatz zur Verfügung.

Nach dem Einschalten des Rechners wird der Benutzer entweder aufgefordert, das in ROM-Form gelieferte erweiterte BASIC oder das HELP-Menü aufzurufen. Nach Drücken der „H" (für HELP)-Taste werden die verschiedenen Optionen dargestellt. Darin sind vornehmlich Input/Output-Angaben enthalten. Der Benutzer kann sowohl von Diskette als auch Cassette Systemprogramme laden oder sich ans „Netz hängen". Außerdem besteht eine Wahlmöglichkeit für die Laufgeschwindigkeit des Cassettenrecorders und die Drucker-Optionen. Es gibt ferner eine sogenannte Front-Panel-Option (im Prinzip ein Speicher-Monitor), mit der der Benutzer die Register des Prozessors und die Speicheradressen einsehen und verändern kann. Mit dem JUMP-Befehl beispielsweise kann die Steuerung an eine Speicheradresse übergeben werden.

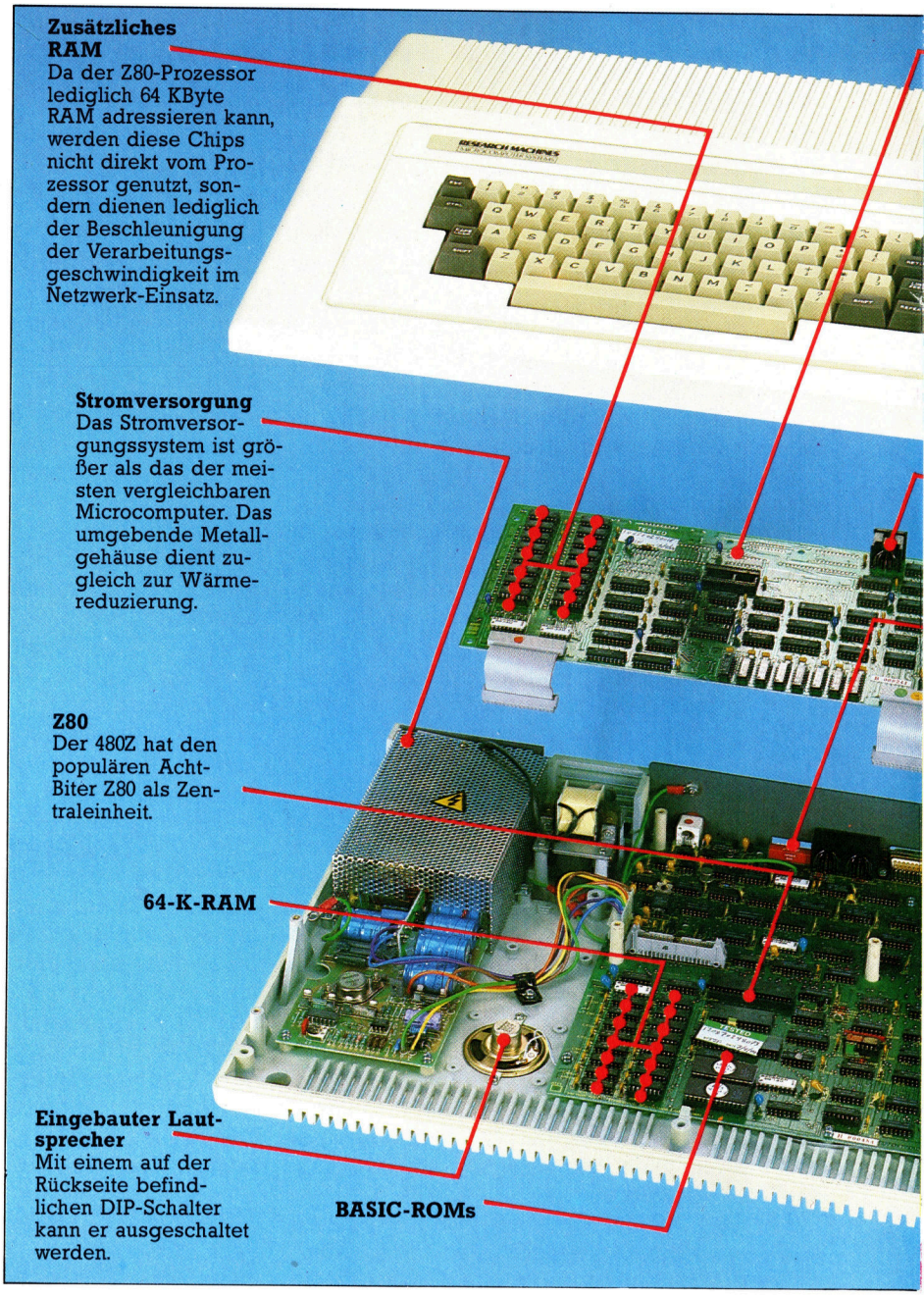
Der 480Z bietet unterschiedliche Bildschirmauflösungs-Modi. Sie reichen von 80 × 25-Textdarstellung bis zur 640 × 192-ultrahoch-Auflösung. Ferner gibt es drei Farb-Modi. Bei mittlerer Auflösung ist die Darstellung von 16 Farben gleichzeitig möglich.

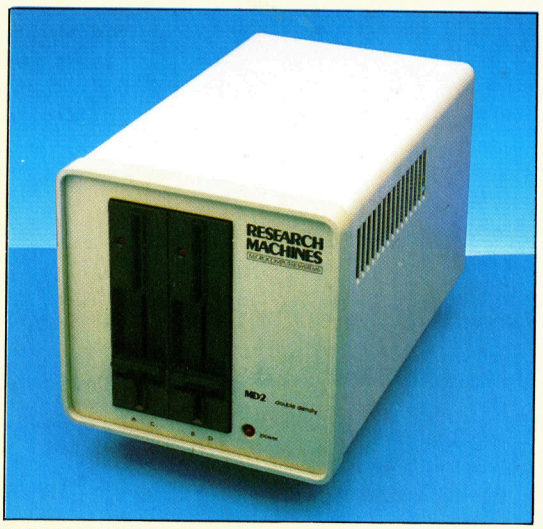
Robust und erweiterbar

Wie beim 380Z dient auch beim 480Z ein Z80 als Zentraleinheit. Dadurch steht eine breite Software-Palette für das System zur Verfügung, einschließlich dem CP/M-Betriebssystem. Die Verfügbarkeit der Software war wohl Hauptgrund für Research Machines, diesen Chip statt eines moderneren Microprozessors zu verwenden.

Im Innern der Maschine ist Platz für erweiternde Chips. Dieser Raum ist zwar nicht mit dem beim 380Z vergleichbar (er ist so konstruiert, daß Erweiterungsplatinen leicht eingeschoben werden können), aber es gibt einen Digital-Analog-Konverter, über den die zusätzliche Schnittstelle mit einem Analog-Gerät verbunden werden kann.

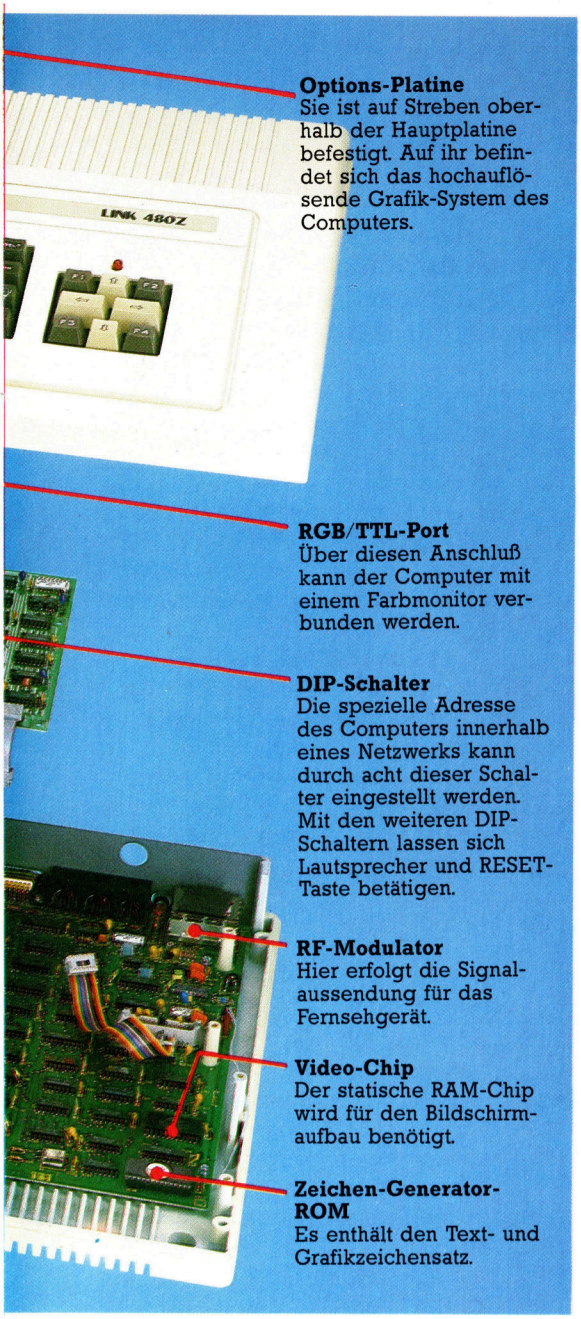
Im Lieferumfang des Computers ist eine Systemdiskette enthalten, auf der sich mehrere Demonstrationsprogramme befinden, ferner eine BASIC-Version mit Disketten-Steuerungs-





Diskettenstation

Jedes der beiden Laufwerke arbeitet doppelseitig, womit insgesamt Zugriff zu vier Seiten möglich ist. Diese werden entsprechend dem CP/M-Aufbau A, B, C und D benannt. Da das Laufwerk mit Disketten in „double density" arbeitet, kann der Rechner die doppelte Datenmenge pro Diskettenseite verarbeiten. Ergänzend hat Research Machines ein Laufwerk mit vierfacher Dichte konstruiert.



Options-Platine
Sie ist auf Streben oberhalb der Hauptplatine befestigt. Auf ihr befindet sich das hochauflösende Grafik-System des Computers.

RGB/TTL-Port
Über diesen Anschluß kann der Computer mit einem Farbmonitor verbunden werden.

DIP-Schalter
Die spezielle Adresse des Computers innerhalb eines Netzwerks kann durch acht dieser Schalter eingestellt werden. Mit den weiteren DIP-Schaltern lassen sich Lautsprecher und RESET-Taste betätigen.

RF-Modulator
Hier erfolgt die Signalaussendung für das Fernsehgerät.

Video-Chip
Der statische RAM-Chip wird für den Bildschirmaufbau benötigt.

Zeichen-Generator-ROM
Es enthält den Text- und Grafikzeichensatz.

befehlen sowie das CP/M-Betriebssystem.

Aus der Gestaltung des Link 480Z ist zu schließen, daß Research Machines einen Computer entwickeln wollte, der die Bedürfnisse von Anwendern im Schulbereich stillt, die eine strapazierbare Allzweck-Maschine benötigen. Unter diesem Gesichtspunkt hat das Unternehmen ein gutes Produkt geliefert. Dennoch ist es enttäuschend, so wenige technische Neuerungen in dem Rechner zu finden. Der Hersteller hat es für wichtiger erachtet, durch die Verwendung des Z80 die Unterstützung durch preiswerte Software zu gewährleisten, als einen modernen 16-Bit-Prozessor einzubauen. Doch zu einer Zeit, da der IBM PC zum Standard für PCs wird, ist die Entscheidung für den Z80 statt des Intel 8088 etwas kurzsichtig.

Schul-Paket

Im Lieferumfang des Link 480Z ist ein Schul-Softwarepaket enthalten. Es gibt insgesamt zwölf Disketten mit hochwertigen Ausbildungsprogrammen.

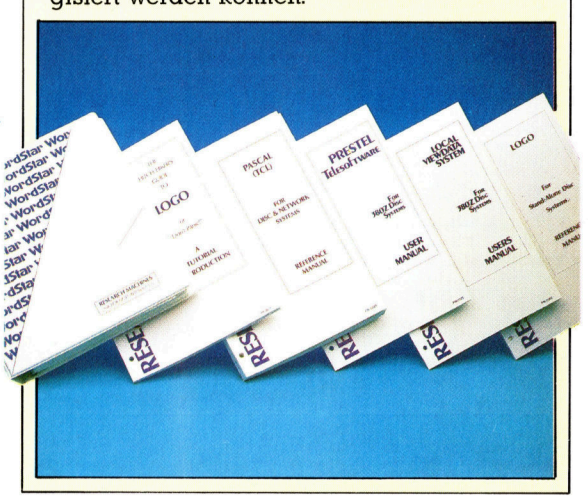
Dazu gehören unter anderem vier Sprachen. Bei SBAS handelt es sich um eine Version strukturierten BASICs, die als hervorragende Implementierung bezeichnet werden kann. Diese Sprache verfügt über eine breite Palette von Kontrollstrukturen für den Programmfluß, darunter auch WHILE... ENDWHILE, CASE... ENDCASE und IF...ENDIF.

Ferner können Prozeduren integriert werden, und globale wie lokale Variablen stehen zur Verfügung. Weiter gibt es eine umfassende PASCAL-Implementierung. Die mitgelieferte Dokumentation ist wie die meisten anderen Research-Machines-Handbücher nicht gerade leicht verständlich, aber detailliert und korrekt. LOGO steht ebenfalls in einer ausgezeichneten Version zur Verfügung, wenngleich die Befehle nicht standardisiert sind.

So wird beispielsweise statt TO der Befehl BUILD verwendet, um Prozeduren zu erzeugen. Es gibt ferner eine teilweise LOGO-Implementierung, ARROW benannt. Wer Maschinensprachenprogramme verwenden will, kann mit dem ZASM arbeiten.

Für die Förderung von Schreib- und Textverarbeitungsfertigkeiten stehen vier verschiedene Programme zur Verfügung. „Touch 'n' GO" ist speziell für Schreibübungen entwickelt worden. WORD dient als Textverarbeitungskurs für Anfänger, mit dem Schüler in die Grundlagen eingeführt werden. WordStar wird ebenfalls mit dem Paket geliefert. TXED ist ein Texteditor, der sowohl für Textverarbeitung als auch für Programmentwicklung genutzt werden kann.

Bei Quest-D handelt es sich um ein Datenbankprogramm, das speziell für die Vermittlung von Grundlagen der Datenspeicherung und des Datenabrufs geschrieben wurde. Weiterführender ist SIR (Schools Information Retrieval), mit dem Schulbibliotheken katalogisiert werden können.



Link 480Z

ABMESSUNGEN
520 × 330 × 80 mm

ZENTRALEINHEIT
Z80, Taktfrequenz 4 MHz

SPEICHERKAPAZITÄT
64 KByte RAM

BILDSCHIRM-DARSTELLUNG
Text: 80 × 25 Zeichen; Mittlere Auflösung: 160 × 192 mit 16 Farben; Hochauflösung: 320 × 192 mit vier Farben; Ultrahohe Auflösung: 640 × 192 mit zwei Farben.

SCHNITTSTELLEN
RF-Anschluß, Monitor, RGB/TTL, Erweiterungsschnittstelle, Cassettenrecorderanschluß, Parallel-Schnittstelle, zwei serielle Schnittstellen, Netzwerk-Videoanschluß.

PROGRAMMIER-SPRACHEN
BASIC, LOGO und PASCAL

TASTATUR
65 Tasten, einschließlich Cursor-Steuerungs- und Funktionstasten.

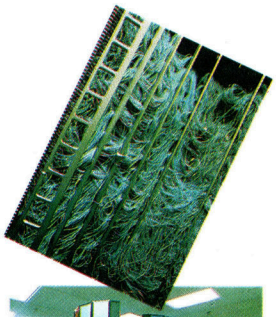
DOKUMENTATION
Die Handbücher sind gut geschrieben und enthalten alle Informationen, die der Anfänger wie auch der fortgeschrittene Benutzer benötigt. Einige Informationen sind jedoch schwer zu finden.

STÄRKEN
Der Link 480Z wurde für die Schule entwickelt. Die CP/M-Fähigkeit und die damit verbundene verfügbare Software – ergänzt um die Netzwerkfähigkeit – machen das System zu einem vielseitigen Schulcomputer. Er ist solide gebaut, womit jahrelange Zuverlässigkeit gewährleistet ist.

SCHWÄCHEN
Es handelt sich um einen technisch überholten Computer, der – verglichen mit Maschinen mit ähnlichen Fähigkeiten – auch zu teuer ist.

Elektronisches Superhirn

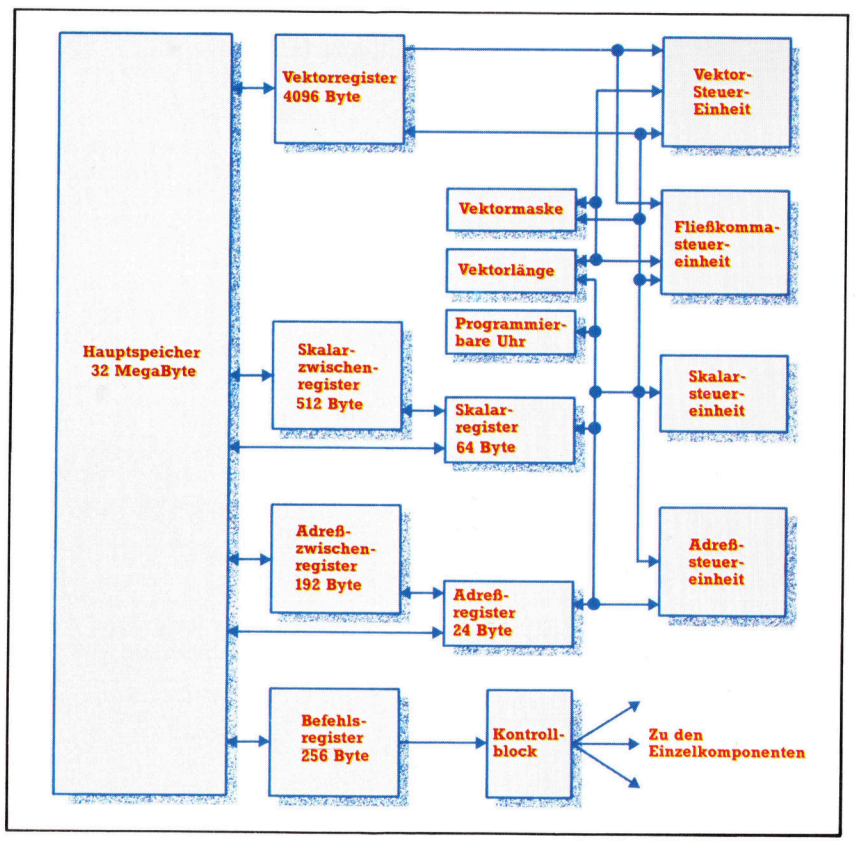
In vielen Anwendungsbereichen benötigt man heute extrem leistungsstarke Computer, die als „Number Crunchers" – Zahlenfresser – bezeichnet werden. Ein Beispiel für diese Supermaschinen ist der Cray-1.



Beim Betrieb unter Steuerung durch einen Front-End-Computer wartet der Cray-1 (Bild oben) mit erstaunlichen Leistungen auf. Das Blockschaltbild ist unten skizziert: Der Supercomputer verfügt über einen Hauptspeicher von 32 MByte und einen 4888-Byte-Registerspeicher. Seine 3400 Einzelplatinen sind durch mehr als 90 Kilometer Kabel miteinander verbunden.

Die Leistung eines Computers ergibt sich, vereinfacht gesagt, aus der Anzahl der Bits, der Datenübertragungsgeschwindigkeit, der Speicherkapazität und der Anzahl der Taktzyklen pro Sekunde. Im Heimcomputer ist die gesamte CPU in einem einzigen Microprozessor vereinigt – beispielsweise dem Z80, dem Intel 8088 und 8086 oder einem Motorola 68000. Die logischen Schaltungen dieser Chip-Typen sind in MOS-Technik (Metall-Oxid-Halbleiter) aufgebaut. Die Daten werden zu jeweils acht oder 16 Bits mit einer Taktfrequenz zwischen 1MHz und 12MHz übertragen und verarbeitet.

Trotz dieser beeindruckenden Zahlen kann ein Heimcomputer bei weitem nicht die riesigen Datenmengen verkraften, die zur Bildsimulation, zur Berechnung des dynamischen Verhaltens von Flüssigkeiten oder für eine Wettervorhersage notwendig sind.



Dazu ein Beispiel: Angenommen, Sie wollten einen Film mit computergestützter Bildsimulation herstellen – jedes Einzelbild mit einer Auflösung von 6000 × 6000 Pixeln, 24 Bildwechsel pro Sekunde. Bei bewegten Bildern muß jeder einzelne Bildpunkt bei jedem Bildwechsel neu berechnet werden – für eine Filmsekunde sind das 864 Millionen Rechenvorgänge. Es kommt noch dazu, daß für jede einzelne Berechnung Dutzende oder gar Hunderte von Befehlen im Maschinencode nötig sind – also viele Milliarden Befehle für eine einzige Filmsekunde. Im Kasten unten rechts vergleichen wir den benötigten Zeitaufwand eines Supercomputer mit der Leistung eines Heimcomputers auf Z80-Basis.

Trotz aller Unterschiede – auch Supercomputer, wie die größten Mainframes oftmals bezeichnet werden, arbeiten nicht viel anders als ein Heimrechner: Befehle und Daten werden aus dem Speicher geholt, die Daten werden den Befehlen entsprechend durch einen Prozessor verarbeitet, und das Resultat wird wiederum gespeichert. Was den „Number Cruncher" vom Heimcomputer trennt, sind hauptsächlich die Datenmenge und die Geschwindigkeit, mit der die Berechnungen ablaufen.

Wir beziehen uns hier auf den von der Firma Cray Research hergestellten Cray-1S/4400 als typischen Vertreter dieser Gattung. Bereits 1976, vier Jahre nach Gründung des Unternehmens, wurde der erste Cray-1 gebaut. Der Rechner – und mit ihm sein Hersteller – gewann schnell einen geradezu legendären Ruf.

Die CPU des Cray-1 befindet sich in einem circa zwei Meter hohen, halbkreisförmigen Gehäuse, das etwa die Form eines Sofas hat. (Stromversorgung und Kühlsystem sind unter den „Sitzen" angeordnet). Die hohe Verarbeitungsgeschwindigkeit wird durch Verwendung bipolarer Halbleiter („normaler" Transistoren, im Gegensatz zu MOS, CMOS, NMOS oder Feldeffekttransistoren) erreicht. Logikschaltung und Speicher des Rechners bestehen aus über 200 000 integrierten Schaltungen auf 3400 einzelnen Platinen. Mehr als 90 Kilometer Leitung wurde verlegt, um die Komponenten miteinander zu verbinden.

Der Cray-1 arbeitet mit 64-Bit-Wörtern (acht-

mal soviel wie der 8-Biter Z80) bei einer Taktfrequenz von 80 MHz (80 Millionen Takte/Sekunde). Eine 64-Bit-Addition dauert nur 37,5 Nanosekunden (Milliardstelsekunden). Zum Vergleich: Die 8-Bit-Addition eines Z80 mit 4 MHz benötigt 1,75 Mikrosekunden (Millionstelsekunden).

Der Hauptspeicher des Cray ist Bestandteil der CPU. Er umfaßt 32 Megabyte, die in 4.194.304 Datenwörtern organisiert sind. Bis zu 2560 Millionen Byte können pro Sekunde übertragen werden.

Zahllose Register

Man kann sich leicht vorstellen, daß der Rechner über eine eindrucksvolle Anzahl von CPU-Registern verfügt. Es gibt 72 Adreßregister à 24 Bit, 72 Skalarregister à 64 Bit und acht Vektorregister à 64 Bit. Der gesamte Registerbereich umfaßt 4888 Byte – der Z80 hat 26.

Zwischen der CPU des Cray und dem Front-End-Rechner liegt eine spezielle Ein/Ausgabesteuerung. Sie sorgt für die Anpassung des schnellen Datendurchsatzes an die besondere Charakteristik des Front-End-Gerätes (IBM, DEC, Data General u. a.). Dieses Subsystem besteht aus bis zu vier Ein/Ausgabeprozessoren, die schon für sich allein einen respektablen Computer darstellen würden.

Unsere Beschreibung der Rechnerkonstruktion kann nicht sehr weit in die Tiefe gehen. Vielleicht vermittelt Ihnen das Blockschaltbild des Cray (linke Seite) aber doch einen kleinen Eindruck vom hohen Standard dieser Wundermaschine. Wir wollen es mit der Feststellung bewenden lassen, daß der Cray-1 ein wirklich extrem leistungsfähiger Rechner ist.

Anwendungsbereiche eines Großrechners

Wettervorhersage
Eine Wettervorhersage gründet sich heute auf weltweit gesammelte Daten, die Bildübertragung über Satelliten und Modellrechnungen. Der Computer kann ein Modell des Wetters nur durch eine riesige Zahl von Berechnungen mit Hunderttausenden von Daten entwickeln. Das Ergebnis soll aber nicht erst in Monaten, sondern nach einigen Stunden bereitstehen – mit dieser Aufgabe werden nur Supercomputer fertig.

Dynamik von Flüssigkeiten
Das dynamische Verhalten von Flüssigkeiten und Gasen ist bei der Konstruktion benzinsparender Autos oder der Planung eines Kühlsystems für einen Atomreaktor ebenso wichtig wie im Flugzeugbau. Jedes Molekül einer Flüssigkeit beeinflußt das Verhalten jedes anderen Moleküls. Um hier Vorhersagen zu treffen, müssen Unmengen von Daten verarbeitet werden – das ist besonders schwierig, wenn die Resultate sofort gebraucht werden. Bei der Berechnung eines Flüssigkeits-Systems kann auf höchste Rechenleistungen nicht verzichtet werden.

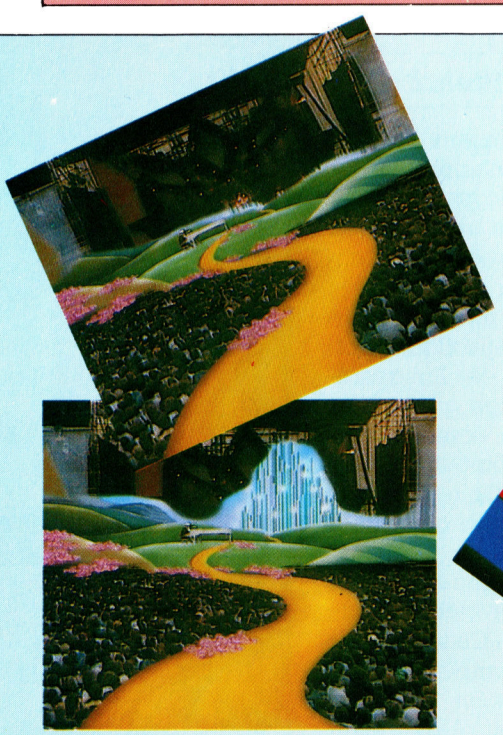
Wirtschaftsprognosen
Modellrechnungen in der Wirtschaft sind außerordentlich komplex, weil – ähnlich wie bei Flüssigkeiten – kleinste Änderungen in einem Teilbereich das gesamte System beeinflussen. Ein ökonomisches „Weltmodell" wird es wohl kaum geben, aber auch vereinfachte Darstellungen machen noch große Schwierigkeiten. Die Großrechner werden auch in diesem Bereich eingesetzt, damit die Resultate schnell verfügbar sind.

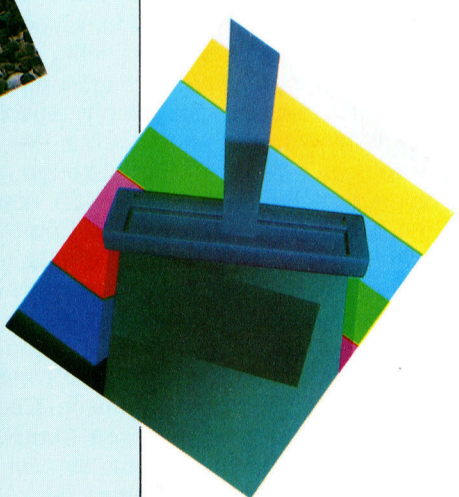
Wettlauf gegen die Zeit

Zum Vergleich der Leistung eines Cray-1 mit der eines Z80-Heimcomputers wie etwa dem Schneider CPC 464 soll der Zeitbedarf für die Produktion einer zehnminütigen Filmsequenz berechnet werden. Die Auflösung soll bei 6000 × 6000 Pixeln liegen, das Bild in einer Sekunde 24mal wechseln. Wir gehen davon aus, daß zur Berechnung eines Pixels beim Z80 100 Maschinenbefehle nötig sind, für die 19 Taktzyklen (4,75 Mikrosekunden) gebraucht werden. Der Cray-1 benötigt durch seine Vektorarithmetik nur 25 Maschinenbefehle (was eher hoch gegriffen ist) mit einer durchschnittlichen Ausführungszeit von vier Taktzyklen (50 Nanosekunden).

Der Z80 braucht
$6000^2 \times 24 \times 60 \times 10 \times 100 \times 4{,}75 \times 10^{-6}$
$= 2{,}4624 \times 10^8$ Sekunden, das sind 7,8 Jahre.

Der Cray-1 kommt mit
$6000^2 \times 24 \times 60 \times 10 \times 25 \times 50 \times 10^{-9}$
$= 6{,}48 \times 10^5$ Sekunden oder 7,5 Tagen aus.





Von Elton Johns „Goodbye Yellow Brick Road" bis zur Gilette-Werbung – überall setzt sich das vom Computer erzeugte Bild durch. Bilder wie diese lassen sich bereits mit Computern mittlerer Größe erzeugen. Es geht aber auch im größeren Maßstab: In „The Last Starfighter" wurden über 20 Minuten des Films von einem Cray-1 hergestellt. Fünfzehn Jahre würde es dauern, bis ein Acht-Bit-Heimcomputer die hierzu nötigen Berechnungen durchgeführt hätte.

Tolle Effekte

In unserer Computer-Galerie zeigen wir faszinierende Grafiken, die auf speziellen Rechnern erstellt wurden.

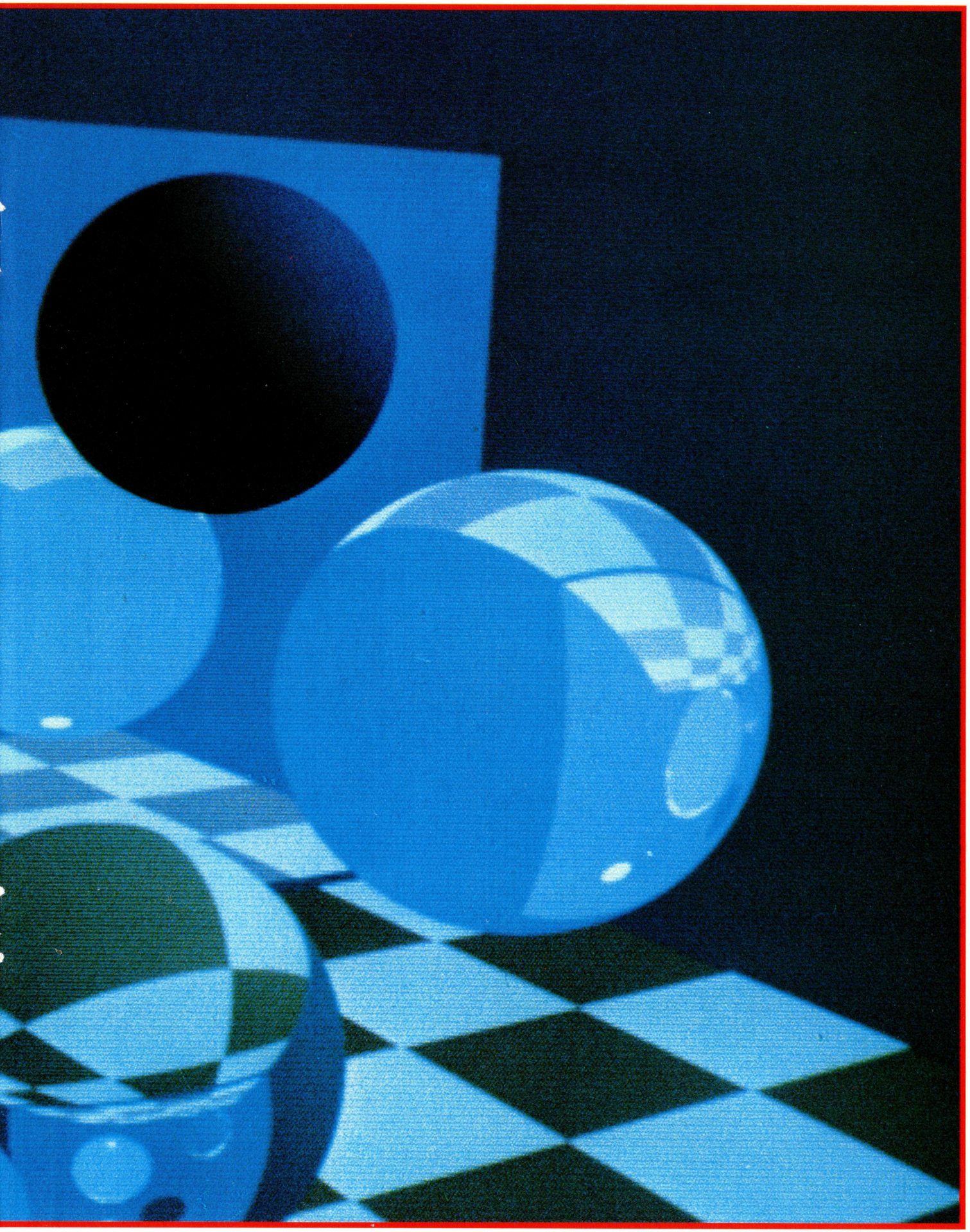
Für viele Computer-Fans, besonders aber für interessierte Laien mit gestalterischen Ambitionen ist der Begriff „Computer-Grafik" zu einem neuen Zauberwort geworden. Brillante Farbabbildungen in Illustrierten und rasante Tricksequenzen in Werbespots oder Spielfilmen wie „Star Wars" und „Tron" haben eine hohe Erwartungshaltung erzeugt. Hat man sich jedoch erst einmal am eigenen Heimcomputer an grafischen Darstellungen versucht, schrumpfen diese Vorstellungen schnell auf ein beschränktes Feld zusammen. So erstaunlich im Vergleich zum Preis die Möglichkeiten der Heimgeräte auch sind – bis hin zu den fotografisch genauen Bildern der „großen Brüder" ist es ein weiter Weg.

Derartige Höchstleistungen sind jedoch nicht für alle grafischen Anwendungsbereiche nötig oder sinnvoll. Für viele Bereiche der Geschäftsgrafik, aber auch beim computerunterstützten Design (CAD = Computer Aided Design), kann man mit kleineren und somit auch kostengünstigeren Maschinen gute Ergebnisse erzielen.

CAD nur menügesteuert

Besonders wichtig sind in diesem Anwendungsbereich auch bedienungsfreundliche Programme: Grafiker und Designer trennen sich nur sehr ungern von ihren traditionellen Arbeitsmitteln wie Pinsel und Bleistift und lehnen Programmierarbeiten völlig ab.

Für diesen rapide wachsenden Anwendungsbereich ist eine gezielte Ausbildung notwendig. Während es jedoch in der Bundesrepublik nur wenige Möglichkeiten gibt, findet man zum Beispiel in den USA eine Vielzahl hochwertiger Lernzentren für Computer-Grafik und -Design. Eine davon ist die Ohio State University, an der auch von D. Lister die hier abgebildete Grafik hergestellt wurde. Bekannt wurde diese Hochschule besonders durch ihren Professor Chuck Csuri, der wiederum als Teilhaber der Firma Cranston/Csuri an der Spitze der Entwicklung von Computer-Grafik steht. Aus diesem Hause stammt auch die „steile 1", die als Kennung der ARD vielen Fernsehzuschauern vertraut ist.



Feldarbeit

In diesem Artikel untersuchen wir den Aufbau der Assemblersprache des 6809 im einzelnen und gehen dabei auf neue Assembleranweisungen und Adressierarten ein.

Ein Assemblerbefehl besteht aus drei Teilen (auch Felder genannt), die jedoch nicht in jeder Zeile enthalten sein müssen:

● **Das Labelfeld** befindet sich in der linken Bildschirmspalte. Labels sind Namen, die Zahlen darstellen und normalerweise Speicheradressen angeben. Sie können eine Länge von einem bis zu sechs Zeichen haben und müssen mit einem Buchstaben anfangen. Labels dürfen Registernamen, Assembler-Op-codes oder anderen, bereits definierten Labels nicht gleichen. Da diese Standardvereinbarungen jedoch nicht strikt festgelegt sind, können einzelne Assembler durchaus anderen Regeln folgen. Ein Leerzeichen am Zeilenanfang gibt an, daß das Labelfeld leer ist. Auch am Ende eines Labelfeldes steht ein Leerzeichen. So ist LABLDA ein gültiges Label, LAB LDA dagegen wird als Label LAB, gefolgt von dem Op-Code LDA, interpretiert.

Der Assembler unterhält einen Speicherzähler, der dem Befehlszähler des Prozessors ähnlich ist. Er enthält die Speicheradresse, an der das nächste Befehls- oder Datenbyte untergebracht werden soll. Trifft der Assembler auf ein Label, legt er dessen Namen in einem Speicherbereich ab, der Symboltabelle heißt und einem BASIC-Array ähnelt. Außer dem Namen wird dort auch die Adresse des Speicherzählers angegeben, die den Punkt bezeichnet, an dem der Assembler dieses Label erstmals gefunden hat. Wird nun der Assembler mit einem Label konfrontiert, sucht er zunächst in seiner Symboltabelle, ob dieses Label bereits existiert. Wenn er es findet, wird das Label gegen diese Adresse ausgetauscht, wenn nicht, speichert er es dort mit dem Inhalt des Speicherzählers.

● **Das Befehls- oder Op-Code-Feld** liegt neben dem Labelfeld. Es enthält ein mnemotisches Kürzel (normalerweise drei Buchstaben) und – falls nötig – einen Registernamen. So besteht ADDA aus dem Kürzel ADD und dem Registernamen A. Der Op-Code stellt den auszuführenden Prozessorvorgang dar. Wie das Labelfeld schließt auch er mit einem Leerzeichen ab.

● **Das Operanden- oder Adreßfeld** enthält Informationen über die Daten, mit denen der Op-Code arbeiten soll. Diese Daten sind entweder Adressen oder Labels, die wiederum Adressen darstellen.

Der Assemblerbefehl

```
LABEL1 ADDA NUM1
```

bedeutet „addiere die Daten, die in der durch das Symbol NUM1 dargestellten Adresse gespeichert sind, in den Akkumulator A". Die Adresse dieses Befehls ist nun als LABEL1 gespeichert. Wenn man auf diesen Befehl verzweigen will, genügt es, das Label als Sprungadresse anzugeben. NUM1 wurde ebenfalls als Label definiert und stellt eine Adresse dar, an der Daten gespeichert sind.

```
CLRA
```

ist ein Beispiel für einen Op-Code, der keinen Operanden benötigt. Er bedeutet „lösche den Akkumulator A – das heißt, setze seinen Inhalt auf Null". Beachten Sie, daß diese Zeile kein Label enthält. Zwar ist der Einsatz von Labels nicht zwingend vorgeschrieben, doch kann mit anderen Anweisungen leichter darauf verzweigt werden. Mit Labels lassen sich auch wichtige Befehle markieren und mit Bemerkungen versehen.

Derart eingesetzte Labels sind jedoch kein vollwertiger Ersatz für ausführliche Kommentare. Bemerkungen lassen sich in jede Zeile eintragen. Es muß nur nach dem letzten Zeichen des Operanden ein Leerzeichen eingegeben werden, dann kann der Kommentar aufgeführt werden. Manche Assembler benötigen Spezialzeichen, um den Anfang des Kommentarfeldes zu kennzeichnen. Ganze Kommentarzeilen werden normalerweise mit einem Stern eingeleitet.

Im Operandenfeld stehen auch die Konstanten in Form von Zahlen oder Zeichenketten. Zahlen werden dezimal interpretiert, wenn sie nicht anders gekennzeichnet sind: durch ein vorangehendes $ oder ein nachstehendes H (beispielsweise $AF08 oder AF08H) als Hexadezimalzahl, durch ein voranstehendes @ oder nachfolgendes Q (etwa @6712 oder 6712Q) als Oktalzahl (Zahlenbasis Acht) oder durch ein voranstehendes % oder nachfolgendes B (beispielsweise %11010011 oder 11010011B) als Binärzahl. Auch der ASCII-Code eines Zeichens kann eine Zahl angeben, wenn ihm ein Apostroph voransteht – 'A bedeutet dann entweder 65 oder $41.

Vielseitig einsetzbar ist auch der aktuelle Wert des Speicherzählers. Zwar ist dieser Wert beim Aufruf des Programms zumeist nicht bekannt, doch ist es möglich, sich im Operandenfeld mit einem Stern darauf zu beziehen. Die meisten Assembler können diese Angabe

auch in einfachen mathematischen Ausdrükken verarbeiten.

```
LDA *+5
```

bedeutet beispielsweise „lade den Akkumulator mit der Speicherstelle, deren Adresse fünf Bytes über dem augenblicklichen Inhalt des Speicherzählers liegt".

Ein Assembler kann normalerweise eine Reihe von Anweisungen oder Pseudobefehlen verstehen, die wie normale Op-Codes in das Programm eingebaut sind. Zwei dieser Befehle haben wir schon früher kennengelernt:

```
CR FCB 13
```

FCB (Fix Constant Byte) reserviert an der aktuellen Speicheradresse ein einzelnes Byte, dem der Wert des Operanden zugewiesen wird. Der Befehl initialisiert eine Speicherstelle (durch das CR-Symbol bezeichnet) mit dem Wert 13.

```
MEMTOP FDB $7FFF
```

FDB (Fix Double Byte) führt den gleichen Vorgang für einen Zwei-Byte-Wert (16 Bits) aus. Speicherplatz kann aber auch ohne Angabe des Inhalts mit dem Pseudobefehl RMB (Reserve Memory Bytes) reserviert werden:

```
TAFEL1 RMB 7
TAFEL2 FCB  $F6
```

Die erste Zeile reserviert für eine Wertetabelle sieben Bytes, deren erste Adresse von dem Label TAFEL1 dargestellt wird. Wenn TAFEL1 daher zum Beispiel die Adresse $C104 kennzeichnet, dann repräsentiert TAFEL2 die um sieben Bytes höhere Adresse $C10B.

Auch ganze Zeichenfolgen lassen sich im Speicher ablegen:

```
ERRMSG FCC 'FEHLER
```

initialisiert sechs Speicherstellen mit den ASCII-Codes für F,E,H,L,E und R. Damit lassen sich auch in die Assemblersprache leicht Meldungen und Promptzeilen einfügen.

Ein weiterer wichtiger Pseudobefehl ist ORG (ORiGin), der den Speicherzähler auf den angegebenen Wert setzt. Er steht am Anfang eines Programmblocks und teilt dem Assembler mit, in welchen Speicherbereich er den Block beim Übersetzen in den Maschinencode ablegen soll. Selbst wenn Assembler eine Standard-Anfangsadresse haben, sollten Sie Ihre Programme immer mit einer ORG-Anweisung beginnen. Ein Programm kann mehrere ORG-Anweisungen enthalten. ORG wird ohne Label oder Operanden eingesetzt.

Die einzige Anweisung, die normalerweise vor dem Pseudobefehl ORG auftaucht, ist EQU (EQUate), die den Variablensymbolen Werte zuordnet. EQU bezieht sich nicht auf den Speicherzähler.

```
RESET EQU $F100
```

definiert das RESET-Symbol und weist ihm den Wert $F100 zu. RESET ist daher die Abkürzung einer Zahl, während ein Label am Anfang einer Befehlszeile immer eine Adresse darstellt, an der Daten oder Code gespeichert sind.

END teilt dem Assembler mit, wo sich das Programmende befindet. Dieser Pseudobefehl nimmt keine Label oder Operanden an.

Ein Maßstab für die Flexibilität von Assemblersprachen sind die Adressierarten – das heißt die Möglichkeiten, Operanden zu interpretieren. Alle bisher aufgeführten Befehle setzten entweder die ganze oder die erweiterte Adressierart ein. Dabei gibt der Wert oder das Label des Operandenfeldes die Speicheradresse mit den Daten an.

Ein-Byte-Adressen

Die direkte Adressierung hat als Operand nur eine Ein-Byte-Adresse, die der Prozessor als niederwertiges Byte einer vollständigen Zwei-Byte-Adresse ansieht. Das höherwertige Byte wird dem Kurzadressierregister entnommen, einem Acht-Bit-CPU-Register, das sich vom Programm ansprechen läßt. Diese Adressierart ist sehr flexibel und läßt sich universell einsetzen: Mit dieser Methode angelegte Subroutinen beziehen sich nicht auf einen festen Speicherbereich. Erst das Laden des Kurzadressierregisters vor Aufruf der Routinen gibt an, in welchem Speicherbereich Daten liegen.

Die erweiterte Adressierung benötigt als Operand eine Zwei-Byte-Adresse. Dieser Befehl bezieht sich immer auf ein festes Speicherbyte und ist daher sehr unflexibel. Der Assembler erkennt die direkte oder erweiterte Adressierung durch die Art des Operanden.

Oft eingesetzt wird auch die unmittelbare Adressierung, bei der die eigentlichen Daten in dem Operandenfeld selbst enthalten sind. Dabei steht vor dem Operanden ein #.

Beachten Sie, daß das Label NUMl die Adresse $1000 darstellt. Sie mögen erwartet haben, daß die ORG-Anweisung auf Adresse $1000 liegt und der darauf folgende Befehl (und damit das Label NUM1) eine höhere

ORG $1000	Anfangsadresse des Maschinencodes ist $1000
NUM1 FCB $FFFF	initialisiert die Speicherstelle NUM1 mit $FFFF
LDD NUM1	lädt $FFFF, den Inhalt von NUM1 in das D-Register
LDD #NUM1	lädt $1000, den unmittelbaren Wert von NUM1, in das D-Register

Adresse hat. Bedenken Sie jedoch, daß ORG nur das Assemblerprogramm anweist, wie es die Übersetzung in den Maschinencode durchführen soll, und nicht Teil des Programms ist. ORG belegt also keinen Speicherplatz. NUM1 nimmt den Wert $1000 an, da der Assembler bei diesem Wert des Speicherzählers zum ersten Mal auf NUM1 trifft. Beachten Sie ebenfalls, daß LDD NUM1 den Inhalt der Speicherstelle NUM1 ($FFFF) in den Akkumulator D lädt.

Nacht der Wölfe

Nach den Erfolgen von Sabre Wulf und Atic Atac stellt die Firma Ultimate einen völlig neuartigen Spieltyp vor: „Knight Lore", mit dreidimensionaler Grafik und exakter Bewegungssteuerung.

Ziel von Knight Lore ist es, ein Objekt zum Schmelzkessel zu bringen, um zu verhindern, daß der Ritter sich auf ewig in einen Werwolf verwandelt. Der Ritter muß zunächst den Zauberer besuchen, um zu erfahren, welche Dinge er dafür überhaupt braucht. Mit dieser Information kann die Suche beginnen. Obwohl die Objekte ihren Platz in jedem Spiel verändern, befinden sie sich immer in den gleichen Räumen.

Ultimate produziert schon seit längerem qualitativ hochwertige Spielprogramme. Darunter gibt es eine Reihe von Labyrinthspielen im Arcadestil, bei denen Spieler in verschiedenartigen Räumen gegen wundersame Gestalten kämpfen müssen, während sie einen Schatz, Teile eines Talismans oder andere mystische Gegenstände suchen, die anschließend zusammenzufügen sind.

Trotz der Beliebtheit dieser Spiele entfernt sich die Firma nun von dem Konzept der zweidimensionalen Arcadeabenteuer wie Sabre Wulf. Knight Lore ist ein völlig neuartiger Spieltyp, der nur wenig Ähnlichkeiten mit früheren Spielen von Ultimate hat. Hier muß ein Ritter das Schloß eines Zauberers aufsuchen und einen Zauberspruch finden, der verhindert, daß er sich auf ewig in einen Werwolf verwandelt. Der Spieler hat vierzig Tage und Nächte für die Lösung zur Verfügung. Die Zeit wird am unteren Bildschirmrand angezeigt, wo sich auch das Fenster mit der „Sonnenuhr" befindet. Tagsüber ist der Spieler (in der Rolle des „Sabre Man") ein normaler Mensch, sobald jedoch der Mond aufgeht, verwandelt er sich, von Krämpfen geschüttelt, in einen Werwolf. Außer während des Verwandlungsvorgangs hat dies glücklicherweise keinen Einfluß auf seine Kampf- und Bewegungsmöglichkeiten. Will man jedoch während der Umwandlung einem Angreifer ausweichen, kann es große Schwierigkeiten geben.

Beim Spielbeginn fällt als erstes die phantastische dreidimensionale Grafik ins Auge. Zwar läßt sich der Blickwinkel nicht wie zum Beispiel bei Ant Attack verändern, doch wird dies bei Knight Lore durch den phantasievollen Aufbau der Räume wieder ausgeglichen.

Knight Lore ist eigentlich kein typisches Arcadeabenteuer. Es besteht nämlich aus einer Reihe logischer Rätsel. In dieser Hinsicht ähnelt es den Abenteuerspielen, bei denen es nur weitergeht, wenn das aktuelle Problem gelöst ist. In Knight Lore sind zwar etliche Räume leer, doch andere enthalten Schwierigkeiten, die vor Verlassen des Raumes überwunden sein müssen. So wird der Keller beispielsweise von kleinen gespenstähnlichen Wesen bewohnt, die auf dem Boden umherflitzen. Wenn sie den Helden berühren, verliert er ein Leben.

Der Schwierigkeitsgrad der Rätsel ist unterschiedlich. Einige sind einfach, andere äußerst kompliziert und wieder andere nur Ablenkungsmanöver. So gibt es beispielsweise einen Raum mit Podesten, über denen große lanzenbewehrte Kugeln hängen. Wenn Ihr Held versucht, über die Podeste zu gehen, wird er verletzt. Die Lösung ist einfach: Lassen Sie ihn um die Podeste herumgehen. Auch die verzerrte Perspektive, die durch eine absichtliche zweidimensionale Darstellung eines dreidimensionalen Objektes entsteht, wird eingesetzt. Blöcke, die auf dem Boden zu ruhen scheinen, schweben oft in der Luft. Fällt der Held von einem dieser Blöcke, verliert er ein Leben und muß den Raum ein zweites Mal durchqueren.

Spielsteuerung

In vielen Räumen müssen Objekte wie Edelsteine, Kelche und Zaubertränke mitgenommen werden. Sie lassen sich einsammeln, indem die Figur darauf springt. Das Objekt erscheint anschließend in der „Inventarliste" in der linken Bildschirmecke. Durch Ziehen des Joysticks läßt sich auch die zusätzliche Höhe gewinnen, mit der man über sonst unüberwindbare Mauern klettern kann.

Der Ritter läßt sich per Joystick oder über die Tastatur in alle vier Richtungen bewegen. Sprünge werden entweder durch Betätigung der dritten Tastenreihe oder des Feuerknopfs ausgelöst. Die Ausrichtung der Spielfigur ist außerordentlich wichtig: Schon ein Sprung in die falsche Richtung kann tödlich sein. Es ist durchaus möglich, daß Sie sich in den ersten paar Spielen zunächst eingewöhnen müssen, da schon eine geringe Bewegung des Joysticks die Blick- und Laufrichtung der Figur verändern kann.

Knight Lore ist ein faszinierendes Spiel mit einem hohen Standard. Selbst jemand, der sich normalerweise nicht mit Rätseln beschäftigt, wird das Spiel interessant finden, da viele Probleme nicht nur einen klaren und wachsamen Geist verlangen, sondern auch schnelle Reaktionen und flinke Finger.

Knight Lore: Für Acorn, Schneider und Spectrum
Herausgeber: Ashby Computer and Graphics Ltd., Ashby de la Zouch, Leicestershire, LE6 5JU
Autoren: Ultimate Play the Game
Joysticks: Wahlweise
Format: Cassette

Mit Glück und Logik

Heute beschäftigen wir uns näher mit strategischer Planung bei Künstlicher Intelligenz und befassen uns sowohl mit Verbesserungen als auch mit alternativen Spielstrategien, wie sie bei Glücksspielen erforderlich sind.

Die in der vorherigen Folge erläuterte Alpha-Beta-Prozedur (das Alpha-Beta-Abschneiden) stellt eine erhebliche Verbesserung direkten „Minimaxens" dar (da somit redundante Zweige des Spielbaums identifiziert und entfernt werden). Dieses Prinzip war über viele Jahre Grundlage der besten Schachprogramme. Kürzlich wurden zwei alternative Strategien entwickelt: einmal der Scout-Algorithmus von Judea Pearl und der B-Star (B*)-Algorithmus von Hans Berliner.

Hauptbestandteil der Scout-Methode ist das Vorhandensein einer fein abgestimmten Evaluations-Funktion, die unlogische Züge ohne weitere Suche abbricht. Nur die erfolgversprechenden Züge werden auch in der Tiefe ausgeführt.

Bei der B*-Methode werden die Züge in der oberen Baumebene überprüft, wobei zugleich versucht wird, so schnell wie möglich zwei Dinge zu tun:

- beweisen, daß der offensichtlich beste Zug tatsächlich der beste Zug ist,
- beweisen, daß keiner der alternativen Züge besser ist.

Anwendungsbeispiele

Diese Doppelstrategie wurde durch das Prozedurenpaar „ProveBest" und „RefuteRest" implementiert. Die Namen beziehen sich auf zwei mögliche Werte an jedem Baumknoten – die eine Prozedur ist eine optimistische Evaluation, die andere eine pessimistische. Ziel ist, den Such-Algorithmus auf jene Bereiche im Spielbaum zu konzentrieren, die unsicher sind und deren Unsicherheit die letztlich zu treffende Entscheidung beeinflussen könnte.

Es wäre falsch anzunehmen, daß die Baumsuche nur beim Computer-Schachspiel Anwendung findet. Es gibt jedoch einige interessante Spiele, in denen diese Such-Philosophie zu scheitern droht – darunter so populäre wie Kartenspiele (vor allem Bridge und Poker) und Brettspiele wie Go und Go-moku.

Diese Spiele können in verschiedenen

Obwohl es einige Spiele gibt, die unter Verwendung von Suchtechniken intelligent von Computern gespielt werden können, indem diese eine Anzahl von Zügen vorhersehen, gibt es viele Spiele, in denen dieses Prinzip nicht funktioniert. Entweder enthält das Spiel einen Zufallsfaktor, so etwa Backgammon, oder der Spielbaum verzweigt sich in unübersichtlichem und unberechenbarem Umfang.

Die Such-Sprache

Spielbaum	Eine Baumstruktur, die sich aus möglichen Zügen ergibt, indem sie gegnerische Züge berücksichtigt.
Halbzugtiefe	(Ply) Eine Ebene im Spielbaum.
Vorausschau	Der Prozeß, durch den ein Spielbaum entsteht.
Rückgestützter Wert	Der einem Knoten zugeordnete Wert in einem Spielbaum, der die darunterliegenden Werte berücksichtigt und rückwärts arbeitet.
Minimaxen	Die Wahl, welcher Wert den Baum weiter gabelt durch Minimieren der ungeraden Halbzugtiefen und Maximieren der geraden Halbzugtiefen.
Alpha-Beta-Algorithmus	Eine Verbesserung des Minimaxens, mit dem jene Baumteile eliminiert werden, die das Ergebnis nicht beeinflussen können.
Verzweigungsfaktor	Die durchschnittliche Anzahl von Verzweigungen oder Zügen in jeder Spielebene. Go hat einen Verzweigungsfaktor von über 200.

Schwierigkeitsgraden gespielt werden. Herausragende menschliche Spieler nennt man berechtigterweise intelligent. Doch alle Versuche, ihre Erfahrungen in die Baumsuche zu integrieren und ein entsprechendes Raster zu erstellen, haben Haken. Einer der Gründe dafür ist, daß der Verzweigungsfaktor einfach zu groß wäre, womit eine so große Anzahl möglicher Züge und Zugkombinationen geschaffen werden würde, daß der Computer sie nicht alle auf einmal abarbeiten könnte.

Hans Berliner, der die B*-Methode entwikkelte, schrieb ein Backgammon-Programm, daß 1980 den Weltmeister in einem Match schlug. Doch dieses Programm sucht überhaupt nicht, zumindest nicht im herkömmlichen Sinne. Vergegenwärtigt man sich, wie Backgammon gespielt wird, wird offensichtlich, daß sein Spielbaum auf der Wahrscheinlichkeitsrechnung basiert. Durch das jeweilige Würfelergebnis werden Zweige eingefügt, die der Kontrolle jedes einzelnen Spielers unterliegen. Somit sind viele der voraussschauenden Implementierungen von vornherein schwer festzulegen.

Aber Berliners Programm kann (besser als ein Mensch) die Bedingungen des Würfelfalls unter verschiedenen Gesichtspunkten berechnen und die sich während des Spiels ergebenden Kombinationen zählen. Zudem verfügt es über hochentwickelte Evaluationsfunktionen.

Go ist ein orientalisches Spiel, das unter Verwendung von Steinen auf einem 18 mal 18 Felder messenden Spielfeld gespielt wird. Ziel dabei ist es, Bereiche des Spielrasters einzukreisen, um Territorium zu gewinnen, und die Steine des Gegners einzukreisen, um sie aus dem Spiel nehmen zu können. Das Zufallselement ist nicht gegeben, doch der Verzweigungsfaktor ist derart groß, daß mit Suchbaum-Techniken keine Lösungen gefunden werden können.

„Go"-Programme betrachten das Spielbrett in Form von Einheiten, die mehr als einen Stein umfassen (wie „Reihen" oder „Armeen") – Elemente, die für das menschliche Auge bedeutungsvoll sind. Ein Grund, warum das Go-Programmieren noch nicht so weit fortgeschritten ist wie die Schachprogrammierung, mag in unserem Verständnis von menschlicher Wahrnehmung liegen.

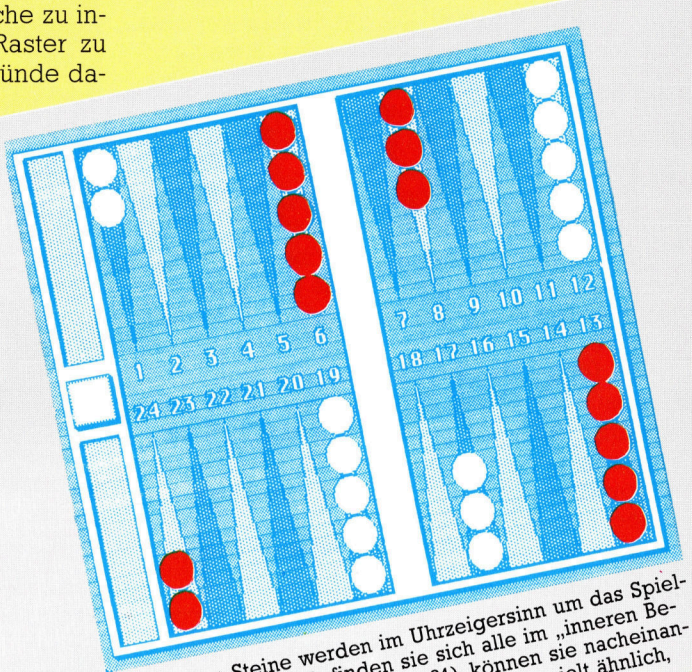
Um das Grundkonzept der Baumsuche zu

Im Innern des inneren Tischs

Backgammon wird auf einem Brett von 24 dreieckigen „Punkten" gespielt, auf dem die Spielsteine entsprechend dem Wurf eines Würfelpaares bewegt werden. Spielziel ist es, mit den eigenen Steinen vor dem Gegenspieler um das Brett herumzukommen.

Man kann entweder einen Stein mit der Gesamtzahl beider Würfel oder zwei Steine mit der Augenzahl der einzelnen Würfel setzen. Bewegt man einen Spielstein entsprechend der Gesamtaugenzahl, wird das als doppelter Zug betrachtet (bestehend aus zwei einzelnen Zügen). Man muß also zwischen Ausgangs- und Endpunkt „setzen".

Punkte (Positionen), die von zwei oder mehr Spielsteinen einer Farbe besetzt sind, dürfen von Steinen des Gegners nicht berührt oder besetzt werden. Befindet sich nur ein Stein an einem Punkt, kann dieser von einem gegnerischen Stein „geschlagen" werden und muß anschließend vom Spielfeld entfernt werden. Die Abbildung zeigt einen Zug im Spiel zwischen Hans Berliners „Backgammon"-Programm und dem Weltmeister Luigi Villa. Die Begegnung fand 1980 in Monte Carlo statt.



Die weißen Steine werden im Uhrzeigersinn um das Spielfeld herumgeführt. Befinden sie sich alle im „inneren Bereich" von weiß (Punkte 19 bis 24), können sie nacheinander aus dem Spiel gewürfelt werden. Rot spielt ähnlich,

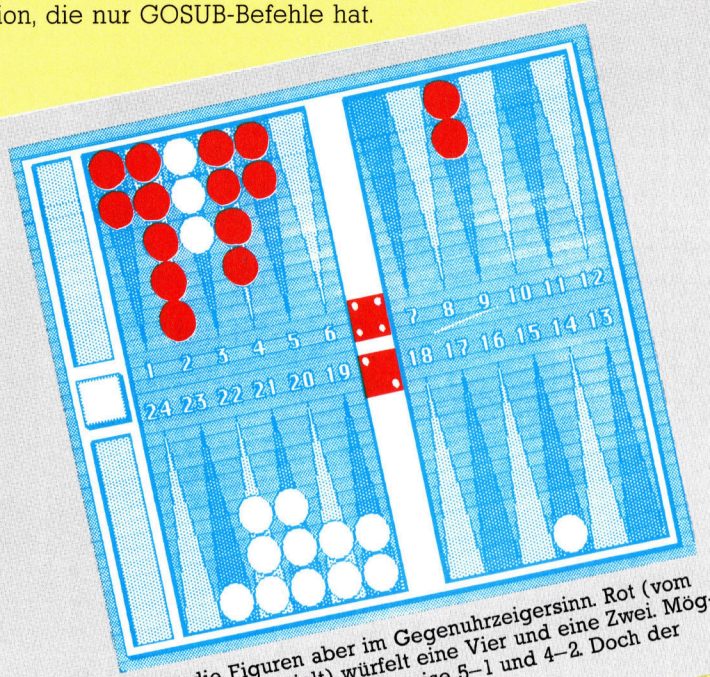
verdeutlichen, zeigten wir ein Spiel, das reine Suchfunktion hatte. Wir stellten es in der letzten Ausgabe als Version für den Acorn B vor.

In diesem Spiel wählen Sie und der Computer abwechselnd eine von vier Funktionen, mit denen ein bestehender Wert zur Erzeugung eines neuen gewandelt wird. Sie müssen versuchen, den Wert auf −255 zu reduzieren, wogegen der Computer versucht, ihn auf 255 zu erhöhen. In jeder Phase bedient sich der Computer des Alpha-Beta-Abschneidens, um zu entscheiden, welche der vier definierten Funktionen für seinen Zweck die richtige ist.

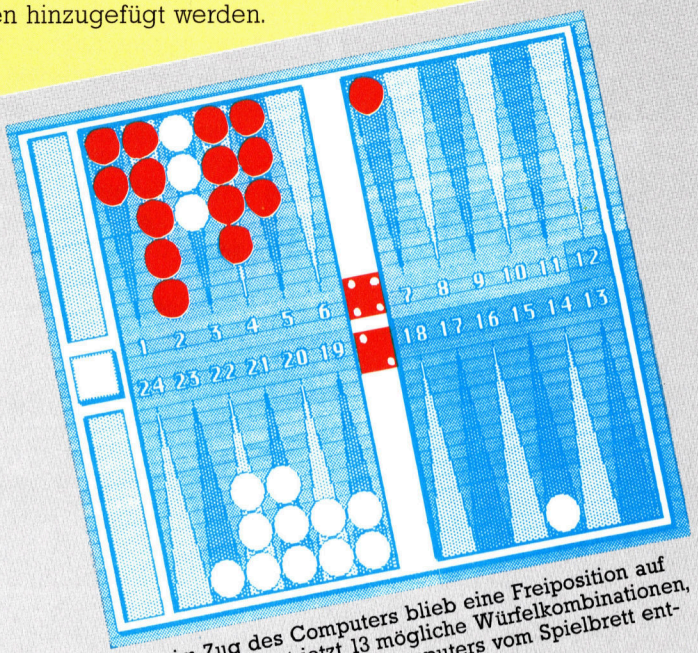
Da es beim C 64 und beim ZX Spectrum im BASIC keine Parameter und lokale Variablen gibt, wiederholen wir das Programm mit einer Version, die nur GOSUB-Befehle hat.

Parameter wie lokale Variablen (mit Ausnahme von D, dem Tiefen-Zähler) haben wir durch Arrays ersetzt, die in Zeile 1100 dimensioniert werden. D zählt hierbei die bereits abgearbeiteten Array-Elemente.

Die beiden Routinen unterscheiden sich vor allem darin, daß alles von „D" verifiziert sein muß, vor allem Array A(), in dem der bisher gefundene beste Wert enthalten ist und B(), in dem sich der schlechteste befindet. Damit wird sichergestellt, daß die verwendeten Alpha- und Beta-Werte den entsprechend richtigen Baumebenen hinzugefügt werden.



führt die Figuren aber im Gegenuhrzeigersinn. Rot (vom Computer gespielt) würfelt eine Vier und eine Zwei. Mögliche Züge sind beispielsweise 5–1 und 4–2. Doch der Computer setzte auf 9–5 und 9–7.



Beim Zug des Computers blieb eine Freiposition auf Punkt 7. Villa hat jetzt 13 mögliche Würfelkombinationen, mit denen der Stein des Computers vom Spielbrett entfernt werden kann.

Das Zahlenspiel

```
80 GOSUB 1000:REM INITIALISATION
90 GOSUB 1600:REM INSTRUCTIONS
100 :
110 REM **** MAIN PROGRAM LOOP ****
120 GOSUB 2000:REM PREPARE NEW GAME
130 INPUT"WHO GOES FIRST (1=YOU,2=ME)";H1
140 IF H1<1 OR H1>2 THEN 130
150 REM *** GAME LOOP ****
160 IF H1=1 THEN GOSUB 3000
170 REM ** PERSON'S TURN **
180 GOSUB 3500:REM BOARD DISPLAY
190 H1=1:REM ALWAYS 1 AFTER 1ST CYCLE
200 GOSUB 4000:REM TEST FOR A WIN
210 IF EG=0 THEN GOSUB 5000
220 REM ** COMPUTER'S TURN **
230 GOSUB 3500:REM DISPLAY GAME STATUS
240 GOSUB 4000:REM TEST FOR END OF GAME
250 IF EG=0 AND M<=33 THEN 150:REM LOOP BACK
260 REM **** FINALE ****
270 GOSUB 6000:REM CONGRATULATIONS
280 INPUT"ANOTHER GAME (1=YES,2=NO)";Y
290 IF Y<1 OR Y>2 THEN 280
300 IF Y=1 THEN 110:REM NEW GAME
310 PRINT:PRINT"SO LONG AND THANKS FOR THE GAME"
320 END
330 :
500 REM **** MAXIMISE ****
510 D=D+1:C1=C1+1
520 IF D>MD OR ABS(V(D))>HI THEN A(D)=V(D):D=D-1:RETURN
530 REM ** ELSE GO DEEPER **
540 P(D)=0
550 REM ** THROUGH TREE **
560 P(D)=P(D)+1:H=P(D):V=V(D):GOSUB 5500:REM MAKE MOVE
570 IF D=1 THEN PRINTCHR$(64+H);" = ";
580 D1=D+1
590 A(D1)=A(D):B(D1)=B(D):V(D1)=V:GOSUB 700:REM CALL MINIMISE
600 IF B(D+1)>A(D) THEN A(D)=B(D+1):K(D)=P(D)
610 IF D=1 THEN PRINTB(D+1);" ";
620 IF P(D)<=3 AND A(D)<B(D) THEN 550
630 IF D=1 THEN BV=A(D):HH=K(D):REM KEEP BEST SO FAR
640 D=D-1:RETURN
650 :
700 REM **** MINIMISE ****
710 D=D+1:C2=C2+1
720 IF D>=MD OR ABS(V(D))>HI THEN B(D)=V(D):D=D-1:RETURN
730 P(D)=8
740 REM ** THROUGH TREE **
750 P(D)=P(D)+1:H=P(D):V=V(D):GOSUB 5500:REM MAKE MOVE
760 D1=D+1:A(D1)=A(D):B(D1)=B(D):V(D1)=V
770 GOSUB 500:REM CALL MAXIMISE
780 IF A(D+1)<B(D) THEN B(D)=A(D+1)
790 IF P(D)<=3 AND B(D)>A(D) THEN 740
800 D=D-1:RETURN
810 :
1000 REM **** INITIALISE ****
1010 BL$=""
1020 REM ** THE FOUR FUNCTION DEFINITIONS **
1030 DEF FNA(X)=2*X-7
1040 DEF FNB(X)=INT(X/2)+1
1050 DEF FNC(X)=4*X+17
1060 DEF FND(X)=3*X-4
1070 LO=-255:HI=255
1080 REM ** ARRAYS USED MY MINIMAX **
1090 D=16
1100 DIM V(D),A(D),B(D),P(D),K(D)
1110 RETURN
1120 :
1600 REM **** INSTRUCTIONS **
1610 PRINT"WELCOME TO THE NUMBERS GAME"
1620 PRINT"I TRY TO MAXIMISE, YOUR JOB"
1630 PRINT"IS TO MINIMISE"
1640 PRINT"TO SEE THE EFFECT OF A MOVE TYPE:"
1650 PRINT"A, B, C OR D. TYPE X TO MAKE IT"
1660 PRINT:RETURN
1670 :
2000 REM **** PREPARATION ****
2010 M=0:V=INT(RND(1)*15)-8:REM INITIAL STATE
2020 EG=0
2030 PRINT"INITIAL STATE = ";V
2040 RETURN
2050 :
3000 REM **** PERSON'S MOVE ****
3010 M=M+1:PRINT
3020 PRINT"YOUR MOVE IS ";
3030 INPUT H$
3040 IF H$="A" THEN PRINT FNA(V):H=1
3050 IF H$="B" THEN PRINT FNB(V):H=2
3060 IF H$="C" THEN PRINT FNC(V):H=3
3070 IF H$="D" THEN PRINT FND(V):H=4
3080 IF H$<>"X" THEN 3020:REM MOVE NOT YET SELECTED
3090 GOSUB 5500:REM MAKE MOVE
3100 RETURN
3110 :
3500 REM **** BOARD DISPLAY ****
3510 PRINT:PRINT"      MOVE";M;" -->";
3520 IF M<1 THEN RETURN
3530 PRINT CHR$(64+H);
3540 PRINT" = ";V:PRINT:RETURN
3550 :
4000 REM **** WIN TEST ****
4010 IF M<1 THEN RETURN
4020 EG=0
4030 IF V<LO THEN EG=-1
4040 IF V>HI THEN EG=1
4050 RETURN
4060 :
5000 REM **** COMPUTER'S MOVE ****
5010 W=V:REM SAVE CURRENT STATUS
5020 M=M+1
5030 MD=6:REM MAX DEPTH
5040 IF M<4 THEN MD=4
5050 IF M>8 THEN MD=8
5060 GOSUB 5200:REM ---> H
5070 V=W:REM RESTORE STATUS
5080 GOSUB 5500 :REM MAKE MOVE
5090 RETURN
5100 :
5200 REM **** MOVE SELECTION ****
5210 BV=LO:D=0
5220 V(1)=V:A(1)=LO:B(1)=HI
5230 GOSUB 500:REM MAXIMISE
5240 H=HH
5250 PRINT:INPUT"PRESS RETURN TO CONTINUE";Q
5260 RETURN
5270 :
5500 REM **** MAKE A MOVE ****
5510 IF H=1 THEN V=FNA(V):RETURN
5520 IF H=2 THEN V=FNB(V):RETURN
5530 IF H=3 THEN V=FNC(V):RETURN
5540 IF H=4 THEN V=FND(V):RETURN
5550 :
6000 REM **** CONGTRATULATIONS ****
6010 PRINT:PRINT"GAME OVER"
6020 IF EG>0 THEN PRINT"I WON IT"
6030 IF EG<0 THEN PRINT"YOU WON IT"
6040 IF EG=0 THEN PRINT"GAME DRAWN"
6050 RETURN
```

Puzzlespiel

Die Technik des modularen Programmaufbaus ist in jeder Programmiersprache nützlich. Sprachen wie PASCAL erleichtern das Arbeiten mit Modulstrukturen, bei BASIC braucht man dafür ein gewisses Durchhaltevermögen. Die Konstruktion von Programmen aus Einzelkomponenten kann hier Hilfestellung geben.

Ein Modul ist eine Befehlsfolge mit einer bestimmten Funktion. Jedes Modul hat exakt definierte Ein- und Ausgänge, seine „Schnittstellen". Was immer zwischen diesen Schnittstellen passiert, sollte vom restlichen Programm vollständig unabhängig sein. Ein komplettes Modul kann als separate Einheit behandelt werden – Daten werden über die Schnittstellen ein- und ausgegeben, was aber im Modul passiert, bleibt verborgen.

Module lassen sich aneinanderreihen, ohne daß der Programmierer sich mit dem „Wie" ihrer Funktion belasten muß. Man kann sehr einfach eine Art Modul-Bibliothek anlegen, aus der man bei Bedarf das Geeignete heraussucht. Und schließlich können Module auch weitergegeben und in fremde Programme eingesetzt werden. Um diese angenehmen Eigenschaften zu nutzen, muß beim Programmieren eines Moduls allerdings sehr genau auf den Datenfluß und den internen Ablauf geachtet werden.

Es gibt eine Grundregel, um sicherzustellen, daß kein Modul „aus der Reihe tanzt" und den Programmablauf stört: Jedes Modul darf nur einen Eingangspunkt und auch nur einen Ausgang haben. Die Befehlsfolge innerhalb des Moduls muß also an einem genau definierten Punkt beginnen und – unabhängig von allen internen Verzweigungen und Schleifen – auch an einem solchen Punkt wieder enden.

Module entsprechen den Algorithmen, die Sie aus den vergangenen Kursabschnitten bereits kennen. In strukturierten Sprachen wie PASCAL können Unterprogramme mit eigenständigen Variablen aufgerufen werden. In diesen Sprachen fällt es relativ leicht, in bestimmte Abläufe (Procedures) zu springen und sie am Ausgangspunkt wieder zu verlassen.

In BASIC läßt sich Ähnliches durch die Kombination von GOSUB. . .RETURN erreichen: Vom Hauptprogramm wird zum Unterprogramm gesprungen, nach dessen Ablauf geht es mit den vom GOSUB-Befehl angesprochenen Zeilen des Hauptprogramms weiter. Allerdings gibt es keine Einschränkungen in bezug auf die Zeile, die das GOSUB-Kommando anspringt. Zwei verschiedene GOSUBs können etwa zu unterschiedlichen Zeilen im selben Unterprogramm führen und von dort mit nur einem RETURN ins Hauptprogramm zurückspringen. Die Resultate können dabei völlig verschieden sein. Auch die Anzahl der RETURNs ist in BASIC nicht festgelegt.

Ohne Selbstdisziplin geht es in BASIC also nicht. Das heißt: Alle GOSUBs zu einem Unterprogramm müssen zur selben Zeile führen, jedes Unterprogramm sollte nur ein RETURN enthalten. Setzen Sie in die erste Zeile jedes Moduls einen REM-Befehl mit dem Namen des Unterprogramms, und verwenden Sie diese Zeile als Einstiegspunkt. Das RETURN steht in der letzten Programmzeile – das muß nicht sein, schafft aber ein Maximum an Klarheit.

Vorsicht bei GOTO

Der GOTO-Befehl kann Programmstrukturen völlig durcheinanderbringen – verwenden Sie ihn also nur mit Vorsicht! GOTO sollte nur zu Zeilen innerhalb desselben Unterprogramms führen, damit nicht versehentlich ein RETURN übersprungen bzw. auf das falsche RETURN reagiert wird. Manchmal soll ein Unterprogramm vor Ausführung aller Befehle verlassen werden. In diesem Fall sollte GOTO zur letzten Unterprogrammzeile führen.

Besonders fehleranfällig ist die Verwendung von GOTO innerhalb von Programmschleifen. Das unkontrollierte Verlassen der Schleife kann dazu führen, daß der gesamte Rest des Programms als Schleife durchlaufen wird. Also: Innerhalb einer Schleife kein GOTO, das aus der Schleife herausführt. Wenn die Schleife vorzeitig verlassen werden soll, muß der Zählindex der Schleife auf seinen Endwert gesetzt werden und GOTO zur Prüfzeile führen (Zeile mit NEXT bzw. WHILE). Man schreibt WHILE und NEXT daher am besten in eine separate Zeile. Programmstrukturen ohne jedes GOTO sind am übersichtlichsten.

Programmabläufe sind speziell bei Verzweigungen schwer kontrollierbar. Eine Entscheidung sollte daher niemals aus einem Unterprogramm herausführen – es sei denn zu einem anderen klar definierten Unterprogramm. Unabhängig von der Verzweigung muß ein Ablauf immer zum festgelegten Endpunkt des Programmteils führen. Das läßt sich am besten anhand eines Flußdiagramms überprüfen. Oft kann die Verwendung von GOTOs in Programmteilen mit Schleifen und Verzweigungen

durch Flags vermieden werden.

Der Datenfluß in ein Modul hinein und wieder heraus geht wie bei den Algorithmen vor sich. Damit die Module unabhängig voneinander arbeiten, darf außer der Datenübertragung zwischen ihnen keine weitere Beeinflussung auftreten. Vom Hauptprogramm werden die Daten zu einem Modul geschickt, von dort sollen die Resultate wieder zum Hauptprogramm transferiert werden.

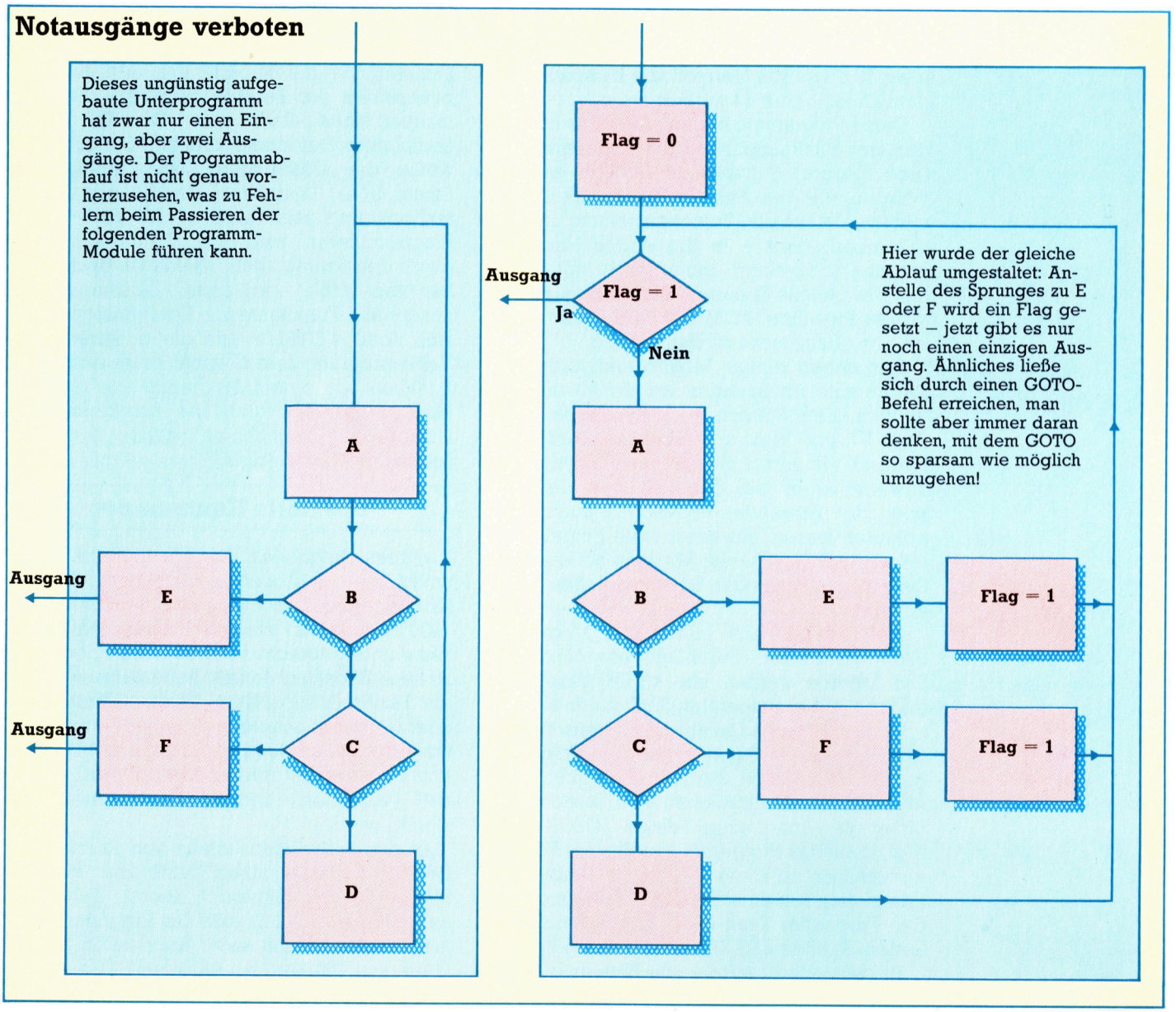
Der Datentransfer innerhalb eines Programms wird über Variablen vorgenommen, deren Bereich („Scope") sich in manchen Programmiersprachen auf Unterprogramme einschränken läßt. In PASCAL müssen etwa die in einem Unterprogramm („Procedure") gültigen Variablen benannt werden. Die „globalen" Variablen des Hauptprogramms gelten überall, während die im Unterprogramm eingesetzten „lokalen" Variablen nur in ihrem speziellen Programmteil Gültigkeit haben.

Lokale Variablen können den gleichen Namen wie globale Variablen haben, ohne daß sich beide gegenseitig beeinflussen. In Programmiersprachen mit lokalen Variablen muß man sich also nicht um differenzierte Benennungen oder die Störung der Variablen durch Variablen aus anderen Programmteilen sorgen. Leider gibt es nur wenige BASIC-Versionen mit lokalen Variablen – wir müssen sie simulieren.

Der einfachste Weg dazu sind Vereinbarungen bezüglich der Variablennamen. Weit verbreitet ist beispielsweise die Verwendung von I, J und K als Schleifenzähler – eine Benennung, die von den Mathematikern übernommen wurde.

Nach der Beschreibung eines Programms in Form eines Flußdiagramms sollte man die einzelnen Unterprogramme numerieren oder auf andere Art benennen. Falls dann Variablen innerhalb eines Unterprogramms lokal eingesetzt werden sollen, hängt man den Namen des Unterprogramms an den Variablennamen.

Notausgänge verboten



Kraftpakete

Vier Kalkulationssysteme für Heimcomputer – Micro Swift, Practicalc II, PS und Vizastar – erheben den Anspruch, daß Heimcomputer durchaus mit größeren kommerziellen Systemen konkurrieren können.

Micro Swift, Practicalc II, PS und Vizastar gehören zu einer neuen Klasse von Kalkulationspaketen, die von den integrierten Systemen Lotus 1–2–3 und seinem Nachfolger Symphony inspiriert wurden. Während Lotus 1–2–3 und Symphony jedoch speziell für den IBM PC und kompatible Maschinen geschrieben wurden (1–2–3 benötigt einen Arbeitsspeicher von mindestens 296 KByte und Symphony 320 KByte), sind die anfangs erwähnten Pakete für Heimcomputer gedacht. Dabei ist es erstaunlich, wie viele Eigenschaften der größeren Systeme die vier Heimpakete in den etwa 30 KByte von Geräten wie beispielsweise dem Commodore 64 unterbringen.

Diese Programme bieten allerdings nur zwei der vier Funktionen, die die umfangreichere (und teurere) Software so attraktiv machen. Würden alle vier Funktionen – Kalkulationssystem, Datenbank, Textverarbeitung und Programmierbarkeit – in die engen Hardwaregrenzen „gezwängt", dann würde mit Sicherheit der gleiche Kompromiß entstehen, der die „Three-Plus-One"-ROM-Software des Plus/4 zu einer Enttäuschung werden ließ.

Wir haben einige Möglichkeiten der vier Programme miteinander verglichen, um ihre individuellen Stärken herauszufinden. Micro Swift, PS und Vizastar sind teilweise programmierbar. Mit dieser nützlichen Fähigkeit – den Tastaturmacros bei Lotus 1–2–3 ähnlich – kann der Anwender Funktionen automatisch ablaufen lassen, die sonst viele Eingaben erfordern würden. Da der Vorgang bei allen drei Systemen verschieden ist, untersuchen wir sie einzeln.

Auf dem PS-Paket lassen sich Module mit BASIC-ähnlichen Befehlen programmieren. Die Module werden mit <f3> gespeichert und mit <U> ausgeführt. Sie können auch automatisch beim Laden ablaufen, wenn sie über SAVE mit einem Punkt hinter dem Programmnamen gespeichert werden. Das Paket besitzt eine Reihe von hilfreichen Fähigkeiten: Wird hinter die Formel eines Feldes GOSUB eingefügt, kann das Programm von diesem Feld aus automatisch zu einer Subroutine verzweigen. Funktionen lassen sich mit FN definieren, und das Programm kann auch Werte von Strings, Zeilen, Spalten und Zahlen mit übergeben.

Bei Micro Swift werden die Befehlslisten einfach in die Spalte Z eingetragen. Der erste Befehl wird dabei durch ein (#) eingeleitet – er gibt den Namen des Programms an. Die letzte Befehlszeile enthält den Befehl @QUIT. Ein einfaches Beispiel:

```
Z1 #SUM
Z2 @SUM(A1,A3)
Z3 @ASSIGN(Z2,A4)
Z4 @QUIT
```

Dieses Programm addiert die Werte der Felder A1, A2 und A3 und ordnet das in Z2 gespeicherte Ergebnis dem Feld A4 zu. Das Modul wird mit #SUM aufgerufen.

Auf Vizastar ist die Programmierung am einfachsten, da die Befehle aus den Anfangsbuchstaben der Befehle bestehen, die sonst manuell eingegeben werden. Zum Aufruf einer bestimmten Datenbank drücken Sie normalerweise die Commodore-Taste, gefolgt von D(ata), U(se), D(atabase), dem Namen der Datenbank und schließlich <RETURN>. Beim Programmieren wird die Commodore-Taste durch den Schrägstrich (/) ersetzt. Beim Drükken von <f8> wird dann /DUDname[RET] ausgeführt. Funktions- und Editiertasten lassen sich über <CTRL> und die entsprechende Taste programmieren, wobei beim Aufruf der Funktion von einem Programm aus das Zeichen dargestellt wird. Die programmierten Cursortasten erscheinen dabei als [up], [down], [left] und [right].

Spezielle Kommandos

Vizastars Datenbank ist außerordentlich vielseitig. In einem Abschnitt der Tabelle, der dem Anwender sonst nicht zugänglich ist (von Zeile 1000 an aufwärts), sind die Formate der Datensätze untergebracht. Jeder Datensatz kann bis zu neun Bildschirmfenster haben, die sich über die Tasten (K)ey, (N)ext, (P)rior, (F)irst, (L)ast oder (C)urrent ansprechen lassen (jeweils der erste Buchstabe des Hauptmenüs). Datensätze können weiterhin Added (hinzugefügt), Replaced (verändert) und ebenso Deleted (gelöscht) werden.

In der Kalkulationstabelle von Practicalc II besteht die Möglichkeit, Texte über ein Feld hinaus in die folgenden leeren Felder zu schreiben. Damit läßt sich das Programm auch als Textsystem mit einer maximalen Zeilenlänge von 100 Zeichen einsetzen. Das System verfügt über Fähigkeiten wie Blöcke verschie-

ben, automatischer Wortumlauf, Einfügen und Löschen.

Sogar Kalkulationstabellen können in die Texte eingefügt werden. Dabei ist die Tabelle noch „aktiv" – das heißt, ihre Formeln, Werte und anderen Inhalte lassen sich im Text verändern, ohne daß das auf Diskette gespeicherte Modell beeinflußt wird.

Obwohl den Programmen durch die engen Speichergrenzen nur wenige Möglichkeiten zur Verfügung stehen, können alle Pakete auf Text- und Datenbankdateien zugreifen, die von anderen Programmen des gleichen Herstellers angelegt wurden. So lassen sich mit Vizastar Textdateien bearbeiten, die mit Vizawrite geschrieben wurden; Micro Swift kann auf Datenbankdateien zugreifen, die mit Micro Magpie erstellt werden; und für Practicalc und PS sind die Dateien von Practicorp und Practifile zugänglich. Da alle Programme sequentielle Formate schreiben, können sie auch untereinander auf Dateien zugreifen und sogar Formate von völlig anderen Programmen wie dem Textsystem Easy Scirpt lesen.

Micro Swift: Für den Commodore 64
Herausgeber: Audiogenic, PO Box 88, Reading, Berks.
Format: Diskette

Practicalc II: Für den 48K Apple II, Acorn B und Commodore 64
Herausgeber: Practicorp, Goddard Road, Whitehouse Ind Est, Ipswitch, Suffolk IP1 5NP
Format: Diskette

PS: Für den Commodore 64
Herausgeber: Practicorp, Goddard Road, Whitehouse Ind Est, Ipswitch, Suffolk IP1 5NP
Format: Diskette

Vizastar: Für den Commodore 64
Herausgeber: Viza Software, 9 Mansion Row, Brompton, Gillingham, Kent ME7 5SE
Format: Diskette mit 4K-Cartridge

Lasten und Lastwagen

Die meisten Transportunternehmen Englands besitzen im Durchschnitt fünf Fahrzeuge – so der Transportberater Terry Palmer. Fast alle Computerprogramme für deren Verwaltung sind jedoch für größere Fahrzeugparks ausgelegt und kosten umgerechnet über 4000 Mark. Der Preis des Paketes von MEM Computing liegt beispielsweise bei 4800 Mark. Dazu kommen weitere 3400 Mark für die Analyse der Fahrzeugkosten. Das Programm kann über 1000 Fahrzeuge verwalten.

Es überrascht nicht, daß nur wenige Inhaber von kleineren Transportfirmen es sich leisten können, ihre Fahrzeuge per Computer zu verwalten – so eine von dem „Science And Engineering Research Council" angeregte und von Terry Palmer durchgeführte Studie. Palmer nahm dies zum Anlaß, ein System zu entwickeln, das sich besser für diesen Abnehmerkreis eignete.

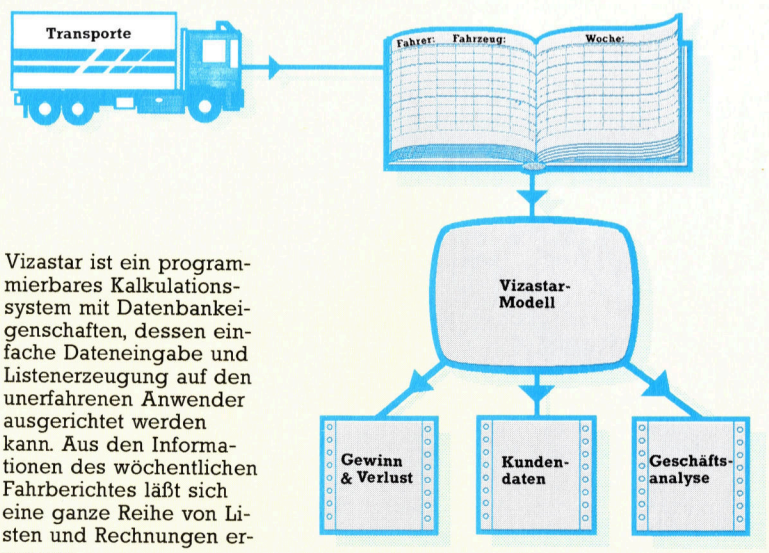
Obwohl er mit Lotus 1–2–3 anfing, lief das Endprodukt schließlich auf einem der meistverkauften Microcomputer – dem Commodore 64. Palmer setzte Vizastar ein, ein programmierbares Kalkulationssystem mit Datenbank, das unter 400 Mark kostet. Er schätzte die Gesamtkosten für Hard- und Software auf circa 4000 Mark – etwa ein Fünftel der Kosten eines größeren Systems.

Palmers Studie war Teil eines Projektes, das er zusammen mit der Polytechnischen Hochschule London ausführte. Darin sollte festgestellt werden, ob kleine Fuhrunternehmen mit der Information, die ein derartiges System bot, etwas anfangen konnten und daher darin investieren würden. Palmer begann mit dem vertrauten Fahrbericht, den alle Fahrer verwenden, und entwickelte Protokollformulare, auf denen abgeschlossene Aufträge, Fahrten, Bestimmungsorte, Treibstoffverbrauch, Barausgaben und Fahrtkosten vermerkt wurden. Am Ende jeder Woche wurden die Daten dieser Formulare in das Kalkulationssystem übertragen.

Schon bei Beendigung der Dateneingabe hatte das System berechnet, ob Gewinn oder Verlust eingefahren wurde, und das Programm erstellte eine vollständige Analyse der vergangenen Woche.

Da Vizastar einen Teil seiner Tabelle auch als Datenbank zur Verfügung stellt, kann man außerdem eine Kundendatei einrichten. Mit dem System können auch Angebote geschrieben werden.

Immer in Bewegung



Vizastar ist ein programmierbares Kalkulationssystem mit Datenbankeigenschaften, dessen einfache Dateneingabe und Listenerzeugung auf den unerfahrenen Anwender ausgerichtet werden kann. Aus den Informationen des wöchentlichen Fahrberichtes läßt sich eine ganze Reihe von Listen und Rechnungen erzeugen.

Objektsammlung

In unserem Abenteuerspiel wird diesmal gezeigt, wie man Routinen entwickelt, die das Aufnehmen und Herumtragen von zahlreichen Objekten ermöglichen.

Im letzten Artikel haben wir uns mit der Befehlsinterpretation und einigen „normalen" Befehlen befaßt. Unter diesen Befehlen befanden sich auch TAKE und DROP sowie die Variationen PICK und PUT. Jetzt können wir die Routinen zur Ausführung dieser Befehle konstruieren. Zuerst untersuchen wir den TAKE-Befehl.

Um die Funktionsweise der TAKE-Routine zu verstehen, wollen wir uns noch einmal die Form verdeutlichen, in der das Programm Objekte innerhalb der Abenteuerwelt organisiert. Im ersten Abschnitt des Projekts entwickelten wir DATA-Anweisungen für jeden Ort. Sie enthalten Ortsbeschreibungen, Namen von Objekten sowie Informationen über mögliche Ausgänge. Nachdem die Daten eingelesen sind, hat das Array IV$(,) folgenden Inhalt:

N	IV$(N,1)	IV$(N,2)
1	GUN	10
2	LAMP	9
3	KEY	5

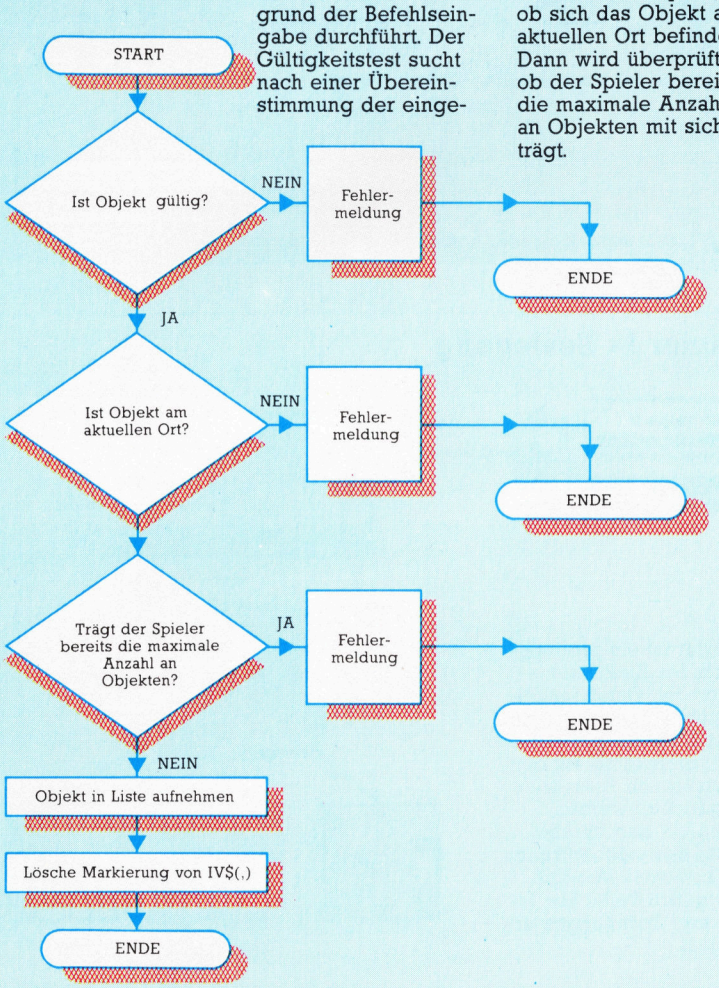
Die erste Spalte des Arrays beinhaltet die Objekt-Namen, in der zweiten Spalte sind die Ortsnummern eines Objekts bei Spielbeginn gespeichert. Während der Beschreibung jedes Ortes wird die zweite Spalte des Arrays überprüft, ob sich an dem Ort irgendein Objekt befindet. Will der Spieler ein Objekt mit dem Befehl „TAKE THE OBJECT" aufnehmen, müssen diverse Bedingungen überprüft werden:

- Ist es ein gültiges Objekt bzw. ist es im Array IV$(,) aufgeführt?
- Befindet sich das Objekt auch wirklich am aktuellen Ort?
- Trägt der Spieler bereits die maximale Anzahl an Objekten, die gemäß Spielregeln gestattet ist?

Wenn alle Bedingungen entsprechend beantwortet werden, kann das Objekt aufgenommen werden. Dazu muß die Objektbeschreibung zum Objekt-Array IC$() hinzugefügt sowie die Positionsmarkierung vom relevanten Eintrag in IV$(,) gelöscht werden. Beachten Sie, daß der Objektname nicht gelöscht wird. Wenn wir eine Positionsmarkierung von −1 für jedes aufgenommene Objekt verwenden, erscheinen solche Objekte bei einer Ortsbeschreibung nicht mehr. Es wäre ja auch unsinnig, die Pistole (GUN) an Ort 10 aufzunehmen, zu Ort 9 zu gehen, wieder zurück zu Ort 10 und dort wieder eine Pistole zu finden. Daher wird in IV$(,) ein Verzeichnis über alle Objekte gespeichert, die zum gegenwärtigen Zeitpunkt vom Spieler nicht transportiert werden. Das Flußdiagramm der TAKE-Routine zeigt die einfache Logik, die befolgt werden muß.

Objekt-Überprüfung

Das Flußdiagramm für die TAKE-Routine zeigt die Vergleiche, die die Routine aufgrund der Befehlseingabe durchführt. Der Gültigkeitstest sucht nach einer Übereinstimmung der eingegebenen Wörter mit den Objektnamen im Inhaltsverzeichnis. Danach wird überprüft, ob sich das Objekt am aktuellen Ort befindet. Dann wird überprüft, ob der Spieler bereits die maximale Anzahl an Objekten mit sich trägt.




```
3700 REM **** TAKE S/R ****
3710 GOSUB 5300:REM IS OBJECT VALID
3720 IF F=0 THEN SN$="THERE IS NO "+W$:GOSUB5500:
     RETURN
3730 OV=F:GOSUB5450:REM CHECK INVENTORY
3740 IF HF=1 THEN SN$="YOU ALREADY HAVE THE "+IV$
     (F,1):GOSUB5500:RETURN
3750 :
3755 REM ** IS OBJECT HERE ? **
3760 IF VAL(IV$(F,2))<>P THEN SN$=IV$(F,1)+" IS
     NOT HERE":GOSUB5500:RETURN
```

```
3770 :
3780 REM ** ADD OBJECT TO LIST **
3790 A=0
3800 FOR J=1 TO 2
3810 IF IC$(J)="" THEN IC$(J)=IV$(F,1):AF=1:J=2
3820 NEXT J
3830 :
3840 REM ** FULL QUOTA **
3850 IF AF=0 THEN PRINT"YOU ALREADY HAVE TWO
     OBJECTS":RETURN
3860 :
3870 SN$="YOU TAKE THE "+IV$(F,1):GOSUB5500
3880 IV$(F,2)="-1":REM DELETE INVENTORY ENTRY
3890 RETURN
```

Betrachten wir nun die drei Tests im Einzelnen. Der wichtigste und komplizierteste ist der Gültigkeitstest. Die einfachste Lösung wäre eine Routine, die den zweiten Teil des Befehls mit jeder Komponente des Arrays IV$(,) vergleicht. Nachteil dieser Methode ist, daß der TAKE-Befehl nur noch in der Form TAKE OBJECT richtig erkannt würde. Selbst Variationen, wie TAKE THE GUN, würden nicht mehr erkannt, da die Routine „THE GUN" anstelle von „GUN" mit dem Inhaltsverzeichnis vergleichen würde. Um dem Spieler hier etwas mehr Freiraum zu geben, müssen wir eine komplexere Methode entwickeln.

Die hierfür naheliegendste Lösung ist, den zweiten Teil des gegebenen Befehls in einzelne Wörter zu unterteilen und diese separat mit dem Inhaltsverzeichnis zu vergleichen. Doch auch hier gibt es Nachteile. Wollen wir etwa für ein Objekt eine Beschreibung mit zwei Wörtern verwenden (wie zum Beispiel „LARGE KNIFE"), könnte das Programm den Befehl „TAKE THE LARGE KNIFE" nicht richtig interpretieren. Die Routine würde die Wörter „THE", „LARGE" und „KNIFE" getrennt vergleichen. Um dieses Problem zu lösen, muß eine Routine entwickelt werden, die jede Objektbeschreibung im Inhaltsverzeichnis nach dem auszuführenden Befehlswort durchsucht. Dabei wird Buchstabe für Buchstabe des Objektnamens verglichen, bis eine Übereinstimmung gefunden oder das Ende der Liste erreicht ist.

Abkürzungen erkennen

Ein Vorteil dieser Methode liegt darin, daß auch Abkürzungen eines Objektnamens erkannt werden. In unserem Beispiel wird der Befehl „TAKE THE KNI" richtig interpretiert, wenn keine anderen Objektnamen außer „LARGE KNIFE" mit der Kombination „KNI" vorhanden sind. Gäbe es einen anderen Namen, so würde eine Interpretation des zuerst gefundenen Eintrags erfolgen. Doch diese Probleme muß man für den Vorteil der größeren Flexibilität in Kauf nehmen. Außerdem lassen sich solche Fehler durch entsprechende Auswahl der Objektnamen umgehen. Müssen zwei Objektnamen dennoch eine identische Gruppe von Zeichen enthalten – BULL und BULLET – sollte der kürzere Name zuerst im Inhaltsverzeichnis eingetragen werden. Ferner sollten unterschiedliche Objektbeschreibungen keine gleichen Wörter enthalten.

Der große Vergleich


```
 10REM **** MATCH AN OBJECT DEMO ****
 40MODE 5:COLOUR 2:DIM V$(3)
 60FOR I=1 TO 3:READ V$(I):NEXT I
 90A$="THE":B$="KNIFE":C$="KNI"
 95S$="                "
110M$=A$:GOSUB 1000:REM MATCH 'THE'
130M$=B$:GOSUB 1000:REM MATCH 'KNIFE'
150M$=C$:GOSUB 1000:REM MATCH 'KNI'
160END
1000REM **** MATCH S/R ****
1010CLS:F=0:LW=LEN(M$)
1030FOR J=1 TO 3:LI=LEN(V$(J))
1042X=1:Y=6:GOSUB2000:PRINT S$
1045X=1:Y=6::GOSUB2000:PRINT V$(J)
1047FOR I=1 TO LI-LW+1
1050X=1:Y=5:GOSUB2000:PRINT S$
1060X=I:Y=5:GOSUB2000:PRINT M$
1070IF MID$(V$(J),I,LW)=M$ THEN F=I:I=LI:J=3
1075FOR D=1 TO 300:NEXT D,I:REM DELAY
1086IF F<>0 THEN GOSUB 2500
1087NEXT J:RETURN
2000REM **** POSITION CURSOR AT X,Y ****
2010PRINT TAB(X,Y);:RETURN
2500REM **** MATCH FOUND ****
2502COLOUR 1:X=F:Y=6:GOSUB2000:PRINT M$
2505FOR K=1 TO 5:X=1:Y=10:GOSUB2000:PRINT S$
2508FOR D=1 TO 500:NEXT:REM DELAY
2510X=1:Y=10:GOSUB2000:PRINT "MATCH FOUND"
2512FOR D=1 TO 500:NEXT D,K:REM DELAY
2520A$=GET$:COLOUR 2:RETURN
3000REM **** INVENTORY DATA ****
3005DATA"SMALL FORK","RED DOOR","LARGE KNIFE"
```

```
  10 REM ** SPECTRUM MATCH **
  40 INK 6:DIM V$(3,20)
1070 IF V$(J,I TO I+LW-1)=M$ THEN F=I:I=LI:J=3
2010 PRINT AT (Y,X);:RETURN
2502 INK 2:X=F:Y=6:GOSUB2000:PRINT S$
2520 A$=INKEY$:IF A$="" THEN 2520
2525 INK 6:RETURN
```

```
  10 REM ** CBM 64 MATCH **
  40 PRINT CHR$(158):DIM V$(3)
  50 DN$=CHR$(17): FOR K=1 TO 5:DN$=DN$+DN$:NEXT:
     DN$=CHR$(19)+DN$
2010 PRINT LEFT$(DN$,Y)TAB(X);:RETURN
2502 PRINTCHR$(28):X=F:Y=6:GOSUB2000:PRINT S$
2520 GETA$:IF A$="" THEN 2520
2525 PRINT CHR$(158):RETURN
```

Die Unterroutine für den Gültigkeitstest in Verbindung mit der TAKE-Routine untersucht den eingegebenen Befehl Wort für Wort, um eine Übereinstimmung mit einem Eintrag im Inhaltsverzeichnis zu finden. Das nebenstehende kurze Programm demonstriert die Funktionsweise der Routine. In diesem Beispiel befinden sich drei Objekte im Inhaltsverzeichnis. Das Programm versucht nun eine Übereinstimmung zu den Wörtern „THE", „KNIFE" und „KNI" zu finden. Ist dies der Fall, wartet das Programm auf einen Tastendruck, um fortzufahren.

```
5300 REM **** VALID OBJECT S/R ****
5310 NN$=NN$+" ":LN=LEN(NN$):C=1:F=0
5315 FOR K=1 TO LN
5320 IF MID$(NN$,K,1)<>" " THEN NEXT K:RETURN
5325 W$=MID$(NN$,C,K-C):C=K+1
5330 LW=LEN(W$)
5335 FOR J=1 TO 3
5340 LI=LEN(IV$(J,1)):REM LENGTH OF OBJECT
5350 FOR I=1 TO LI-LW+1
5360 IF MID$(IV$(J,1),I,LW)=W$ THEN F=J:I=LI:J=3:K
=LN
5370 NEXT I,J,K
5380 RETURN
```

Wurde eine Übereinstimmung gefunden, wird die Variable F auf den Wert des Elementes gesetzt, der dem Objekt im Befehl entspricht. Findet das Programm kein Objekt, ist der Wert von F Null, um anzuzeigen, daß kein derartiges Objekt im Spiel vorkommt.

Ist die Array-Nummer des Objekts einmal bestimmt, kann die Position des Objekts mit der Ortsvariablen P verglichen werden. Das aufzunehmende Objekt befindet sich in IV$(F,1) und seine Position ist in IV$(F,2) gespeichert. In Zeile 3760 der TAKE-Routine von Haunted Forest wird dieser Wert mit P verglichen. Die Fehlermeldung – „OBJECT is not here" – ist jedoch nicht immer korrekt. Das Objekt kann ja schon vom Spieler mitgenommen worden sein. Daher sollte vor Ausgabe der Fehlermeldung noch das Inhaltsverzeichnis des Spielers überprüft werden. Ist das Objekt darin vorhanden, sollte eine andere Fehlermeldung wie etwa „You already have the OBJECT" ausgegeben werden. Die folgende Unterroutine überprüft das Hauptinhaltsverzeichnis und setzt das Flag HF auf 1, wenn der Spieler das Objekt bei sich trägt. Diese Kondition wird durch −1 im relevanten Element des Arrays angezeigt.

```
5450 REM **** IS OBJECT HELD S/R ****
5460 HF=0
5470 IF IV$(OV,2)="-1" THEN HF=1
5480 RETURN
```

Maximale Belastung

Die Überprüfung der Anzahl der vom Spieler getragenen Objekte und das Hinzufügen eines Objekts können zusammen ausgeführt werden. Unter Verwendung von IC$() kann mit einer FOR. . .NEXT-Schleife freier Platz für einen neuen Eintrag gesucht werden. Da bei Haunted Forest nur zwei Objekte gemeinsam getragen werden dürfen, wird die FOR. . .NEXT-Schleife nur zweimal ausgeführt. Wird kein freier Platz gefunden, erscheint eine Meldung.

Die letzte Aufgabe ist, die Positionsmarkierung des aufgenommenen Objekts zu entfernen. Dies wird mit IV$(F,2)=−1 erreicht.

Jetzt, da der Spieler Objekte aufnehmen kann, wird ein weiterer Befehl sinnvoll. Oft ist es nützlich, wenn der Spieler sehen kann, welche Objekte er mit sich trägt. Angenommen, der Spieler kommt zu einer verschlossenen Tür und hat vergessen, daß er 20 Spielzüge zuvor einen Schlüssel aufgenommen hat. Mit dem LIST-Befehl kann er schnell überprüfen, ob er den Schlüssel bei sich trägt. Der erforderliche Programm-Code besteht aus einer einfachen FOR. . .NEXT-Schleife, die den Inhalt von IC$() darstellt.

```
4100 REM **** LIST CARRIED INVENTORY ****
4110 PRINT"OBJECTS HELD:"
4120 FOR I=1 TO 2
4130 PRINT"  ";IC$(I)
4140 NEXT I
4150 RETURN
```

Digitaya Listings

```
2140 REM **** TAKE S/R ****
2145 IV$(4,1)="TICKET TO TRI-STATE"
2150 GOSUB5730:REM IS OBJECT VALID
2160 IF F=0 THEN PRINT"THERE IS NO ";W$:RETURN
2170 REM ** IS OBJECT ALREADY TAKEN ? ****
2180 OV=F:GOSUB5830
2190 IFHF=1 THEN SN$="YOU ALREADY HAVE THE "+IV$(F
,1):GOSUB5880:RETURN
2200 :
2210 REM ** IS OBJECT HERE **
2220 IF VAL(IV$(F,2))<>P THENSN$=IV$(F,1)+" IS NOT
 HERE":GOSUB5880:RETURN
2230 :
2240 REM ** ADD OBJECT TO LIST **
2250 AF=0:FOR J=1TO4
2260 IFIC$(J)=""THENIC$(J)=IV$(F,1):AF=1:J=4
2270 NEXTJ
2280 :
2290 REM ** CHECK FOR FULL QUOTA **
2300 IF AF=0THENPRINT"YOU ALREADY HAVE 4 OBJECTS":
RETURN
2310 :
2320 SN$="YOU TAKE THE "+IV$(F,1):GOSUB5880
2330 IV$(F,2)="-1":REM DELETE POSITION ENTRY
2340 RETURN

5730 REM **** VALID OJECT S/R ****
5740 NN$=NN$+" ":LN=LEN(NN$):F=0:C=1
5745 FOR K=1 TO LN
5750 IF MID$(NN$,K,1)<>" " THEN NEXTK:RETURN
5755 W$=MID$(NN$,C,K-C):C=K+1:LW=LEN(W$)
5760 FORJ=1 TO 8
5770 LI=LEN(IV$(J,1)):REM LENGTH OBJECT
5780 FORI=1TO LI-LW+1
5790 IFMID$(IV$(J,1),I,LW)=W$THENF=J:I=LI:J=8:K=LN

5800 NEXT I,J,K
5810 RETURN
5820 :
5830 REM **** IS OBJECT HELD S/R ****
5840 HF=0
5850 IFIV$(OV,2)="-1"THEN HF=1
5860 RETURN

2540 REM **** LIST INVENTORY S/R ****
2550 PRINT"OBJECTS HELD:"
2560 FORI=1TO4
2570 PRINT"  ";IC$(I)
2580 NEXTI
2590 RETURN
```

BASIC-Dialekte

Spectrum:
Führen Sie die folgenden Änderungen am Listing von Haunted Forest durch:

Ersetzen Sie SN$ durch S$, IV$(,) durch V$(,), IC$() durch I$() und NN$ durch R$.

```
5320 IF R$(K TO K)<>""THEN NEXT K:RETURN
5325 LET W$=R$(C TO K−1)
5360 IF V$(I TO I+LW−1)=W$ THEN LET F=J:LET
     I=LI:LET J=3:LET K=LN
```

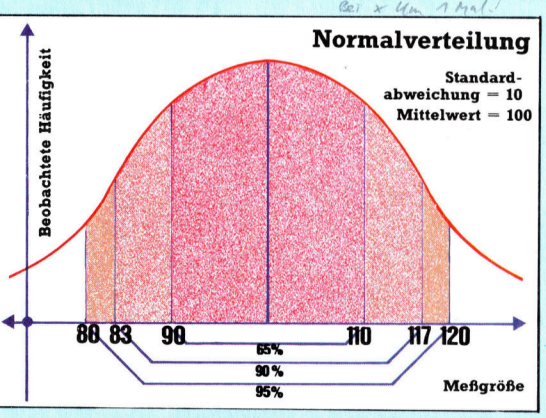
Beim Digitaya-Listing ersetzen Sie dieselben Variablennamen. Führen Sie außerdem die gezeigten Änderungen entsprechend in den Programmzeilen 5750, 5755 und 5790 durch.

Fachwörter von A bis Z

Hier werden einzelne Fachausdrücke eingehend behandelt. Da bei der Kommunikation mit dem Computer meist die englische Sprache verwendet wird, werden hier zunächst die englischen Begriffe genannt, dann die deutsche Übersetzung. In den Gesamtindex werden sowohl deutsche als auch englische Stichwörter aufgenommen, damit Sie es leichter haben, das von Ihnen Gesuchte zu finden.

Frequency = Frequenz, Häufigkeit
Das englische Wort „Frequency" steht sowohl für die physikalisch/technische Größe „Frequenz" wie für den statistischen Begriff der „Häufigkeit" bestimmter Eigenschaften. Die Frequenz eines periodischen Vorgangs gibt an, wie oft ein bestimmter Ablauf – etwa eine elektrische Schwingung – innerhalb der Zeiteinheit stattfindet. Die Frequenz wird in Hertz, abgekürzt Hz, als Anzahl der Zyklen pro Sekunde angegeben. Die Wechselspannungsfrequenz in unserem Versorgungsnetz beträgt 50 Hz; die Musiknote A über dem mittleren C hat 440 Hz; der Prozessor Z80A arbeitet im allgemeinen mit einer Taktfrequenz von 4,25 MHz (entsprechend 4 250 000 elementaren Operationen pro Sekunde), und den NDR II empfangen Sie in Hamburg auf UKW bei 87,6 MHz. Bei der Ausbreitung von Schwingungen in Form von Licht- oder Schallwellen hängt die Frequenz f mit der Wellenlänge λ (Abstand benachbarter identischer Schwingungszustände, etwa zweier Wellenberge) und der Ausbreitungsgeschwindigkeit c zusammen: $c = \lambda * f$

Da die Ausbreitungsgeschwindigkeit elektromagnetischer Wellen (z. B. sichtbares Licht, Infrarot und Rundfunkwellen) circa 300 000 km/s beträgt, errechnet sich für die NDR-Frequenz beispielsweise eine Wellenlänge von 3,42 m.



Von allen Häufigkeitsverteilungen ist die Gauß- oder Normalverteilung in Theorie und Praxis am meisten anzutreffen. Wegen ihrer charakteristischen Form wird die Normalverteilung auch als „Glockenkurve" bezeichnet.

Frequency Distribution = Häufigkeitsverteilung
Wenn Versuchsresultate grafisch so dargestellt werden, daß auf der x-Achse eine veränderliche Meßgröße und auf der y-Achse die zugehörige Häufigkeit aufgetragen wird, ergibt sich ein Bild der „Häufigkeitsverteilung". Dabei muß man die Daten im allgemeinen in Untergruppen zusammenfassen, um eine brauchbare Verteilung zu erhalten. Wenn es sich zum Beispiel um die Häufigkeit gemessener Körpergrößen handelt, lassen sich Gruppen von 161–165 cm, 166–170 cm, 171–175 cm usw. bilden. Die gewonnenen Verteilungen lassen sich mathematisch analysieren und ermöglichen beschreibende Aussagen und die Erstellung von Prognosen.

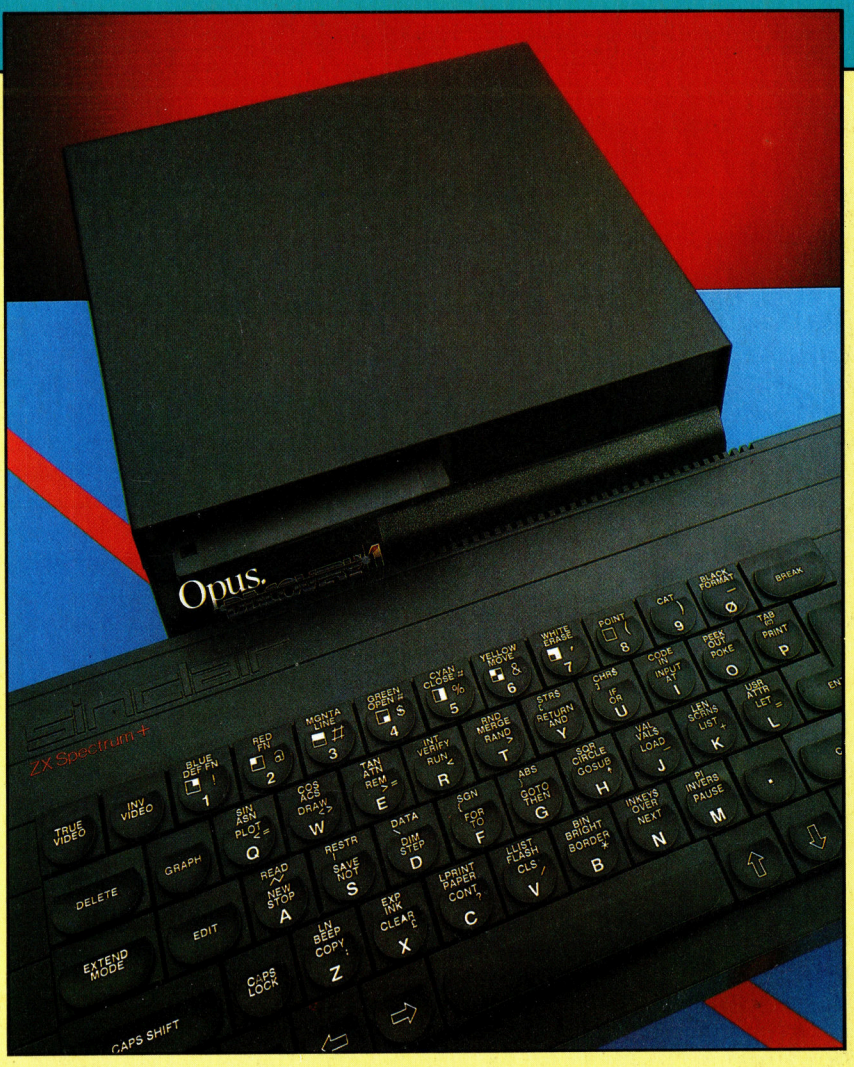
Häufig werden auch die Begriffe Gauß- oder Normalverteilung erwähnt. Viele menschliche Eigenschaften sind über die Bevölkerung normalverteilt, z. B. Körpergröße, Augenfarbe und (angeblich) auch die Intelligenz. Stichprobenverteilungen nähern sich vielfach der Normalverteilung, wenn der Probenumfang groß genug wird.

Die einzelne Stichprobe ist durch den Mittelwert („durchschnittlicher" Wert der Meßgröße) und die „Standardabweichung" (ein Maß für die Streuung der Daten um den Mittelwert) gekennzeichnet. Bei einer Normalverteilung liegen etwa 65% der Daten weniger als eine Standardabweichung vom Mittelwert entfernt und mehr als 90% weniger als zweieinhalb Standardabweichungen. Die ganze mathematische Statistik fußt größtenteils auf der Analyse von solchen Häufigkeitsverteilungen.

Full Duplex = Vollduplex
Eine Telefonverbindung arbeitet „vollduplex", wogegen der Funksprechverkehr i. a. „halbduplex" abläuft. Der Vollduplexbetrieb läßt eine Datenübertragung in beiden Richtungen gleichzeitig zu, das Halbduplexverfahren erlaubt das nur im Wechsel.

Fuzzy Theory = Fuzzy-Theorie
In digitalen Systemen gibt es nur eindeutige Zustände wie Eins oder Null, Ja oder Nein. Diese elektronikbedingte zweiwertige Logik hat auch die modellmäßige Behandlung der realen Welt in den Rechnerprogrammen geprägt. Damit kommt beim computerorientierten Denken zu kurz, was sonst dem Menschen sehr von Nutzen ist – nämlich die Fähigkeit, mit Wahrscheinlichkeiten und Unsicherheitsfaktoren umzugehen und aufgrund unvollständiger Informationen Entscheidungen zu fällen. Die Fuzzy-Theorie (engl. fuzzy = verschwommen) versucht, dieses Prinzip auf den Rechner zu übertragen. Sie operiert mit einer vielwertigen Logik, in der eine Aussage richtig, wahrscheinlich richtig, vielleicht richtig, wahrscheinlich falsch und falsch sein kann. Dies führt zu unerwarteten Einsichten und bizarren Konstruktionen wie den Fuzzy-Mengen und Fuzzy-Relationen. Die weiteren Forschungsarbeiten über Künstliche Intelligenz dürften auf diesem Gebiet noch interessante Entwicklungen bringen.

Bildnachweise
1261, 1262, 1263, 1269: Chris Stevens
1264: Liz Heaney
1265, 1278, 1285: Ian McKinnell
1266, 1267: Kevin Jones
1273: CAL-Video für Mike Mansfield Enterprises und für BBDO
1274, 1275: D. Lister
1279: David Lawrence
1283: Liz Dixon
1286: Gary Capps-Jenner



Unter der Bezeichnung „Discovery“ vertreibt die Firma Opus Diskettenlaufwerke für Sinclair-Computer. Gedacht sind die Geräte als Konkurrenz zu Wafadrive und Microdrive. Wir untersuchen, ob sich das Diskettenlaufwerk als Massenspeicher für den Spectrum eignet.

+ Vorschau +++ Vorschau +++ Vorschau +++ Vorschau +++ Vorschau +

computer kurs

Heft 47

Recursionen

Diese BASIC-Folge beschreibt eine Technik, die bei fortgeschrittenen Programmen eingesetzt wird: die Recursion.

Experten-Systeme

In der Serie „Künstliche Intelligenz“ geht es um hoch strukturierte Programme, die Experten-Systeme.

Ausdruck

Der PROLOG-Kurs erklärt die Strukturelemente „Ausdruck“ und „Tatsache“.

Klassenbester

Zukunftsweisend zeigt sich der 16-Bit-Rechner „Nimbus“ von Research Machines.