Einsteigen - Verstehen - Beherrschen

DM 3,80 öS 30 sfr 3,80

computer kurs

Heft 67

Veteran: Spectravideo

Der Modellentwurf

Hilfsroutinen zum Zeichnen

Gestatten, Dr. LOGO

Ein wöchentliches Sammelwerk

computer kurs

Heft 67

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.

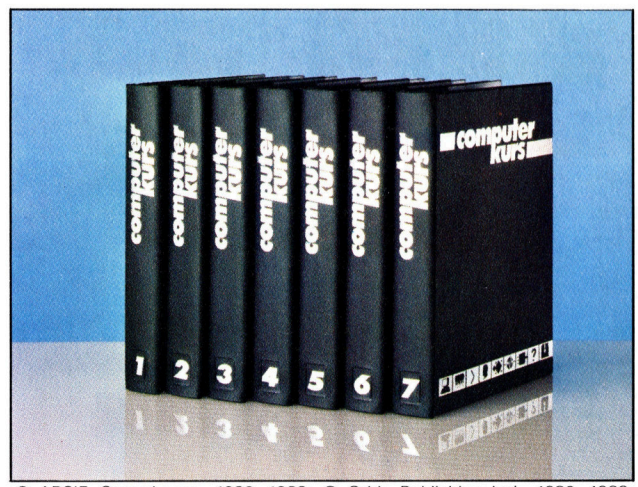
Schweiz: Der Sammelordner kostet sfr 15. Bitte wenden Sie sich an Ihren Kiosk; dort werden Sie jederzeit die gewünschten Exemplare erhalten.

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), Peter Aldick, Holger Neuhaus, 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

Modell-entwurf

In unserer Serie über Künstliche Intelligenz sprachen wir vielfältige Anwendungsmöglichkeiten, aber auch den hohen Preis von Expertensystemen an. In diesem Abschnitt betrachten wir ein Programmpaket, mit dem sich auch Laien ein Expertensystem nach Maß schneidern können.

Expert-Ease ist ein Generator für Expertensysteme aus dem Hause Thorn-EMI. Mit dem Programm können Systeme für die unterschiedlichsten Zwecke schnell und relativ einfach selbst aufgebaut werden.

Wie der Name schon andeutet, werden Expertensysteme in eng umgrenzten Fachgebieten eingesetzt, etwa für medizinische Diagnostik oder geologische Feldversuche. Ein Generator für Expertensysteme bietet dagegen die Möglichkeit, sie auch in Gebieten einzusetzen, für die ein kommerzielles System zu teuer wäre. Ein Beispiel für einen solchen Ansatz ist das Programm Expert-Ease von Thorn-EMI Software. Damit kann jeder sein eigenes Expertensystem (ein sog. „Modell") erzeugen. Das Programm läuft beispielsweise auf dem IBM PC und ACT Sirius mit mindestens 128 KBytes RAM.

Wir erläuterten bereits, daß die eigentliche Arbeit im Kern des Expertensystems abläuft – im „Folgerungsmodul." Dieser Programmteil zieht logische Schlüsse. Die Programmierung dieses Moduls ist so schwierig (und damit teuer), daß Expertensysteme für private Anwender bisher unerschwinglich waren. Eben diese Programmierung übernimmt nun Expert-Ease selbst: Anstelle des Fachmanns benötigen wir nur noch die Software und natürlich den geeigneten Computer.

Vergleichende Konzepte

Für die Konstruktion eines Folgerungsmoduls gibt es verschiedene Ansätze: Regelhierarchien, die auf IF-THEN-Schlußfolgerungen beruhen und semantische Netze, die mit Mengen und Untermengen arbeiten (so ist etwa „Batterie leer" ein Element der Menge „Elektrische Fehler"). Andere Verfahren arbeiten wie eine gewöhnliche Datenbank – mit dem Unterschied, daß aus den Feldeintragungen auch Schlußfolgerungen gezogen werden. Expert-Ease benutzt eine andere Methode: ein „lernfähiges System vergleichender Konzepte" (Analogue Concept Learning System-ACLS).

ACLS wurde am Laboratorium für Künstliche Intelligenz der Universität Edinburgh unter Beteiligung der Intelligent Terminals Ltd. entwickelt. Es handelt sich dabei um eine Weiterentwicklung des „Iterative Dichotomisers 3" (Iterativer Entscheider), ein Public-Domain-Programm in PASCAL.

Mit Expert-Ease wird ein Modell im „Rückwärtsgang" aufgebaut. Man muß sich also erst entscheiden, was die möglichen Schlußfolgerungen sind. Erst dann kann festgelegt werden, welche Fragen dem Anwender gestellt werden und was für Schlußfolgerungen aus diesen Antworten gezogen werden sollen. Der Aufbau von Expert-Ease soll an einem einfachen Beispiel vorgeführt werden – an der Frage nämlich: Geh' ich heute zum Fußballspiel oder nicht?

Die Alternativen bei diesem Problem sind offensichtlich nur „Gehen" oder „nicht gehen". Damit sind wir schon bei der Frage, wovon unsere Entscheidung abhängt. Um es einfach zu machen, setzen wir einmal voraus, daß in die-

Beispiel/ Frage	Gutes Wetter	Genug Zeit?	Entscheidung
1.	JA	JA	HINGEHEN
2.	JA	NEIN	NICHT
3.	NEIN	JA	NICHT
4.	NEIN	NEIN	NICHT

sem Fall nur das Wetter und unsere Zeit eine Rolle spielen. Dann würde sich unsere erste Entscheidungstabelle ergeben. Wir gehen also nicht, wenn auch nur eine Frage mit „nein" beantwortet ist. Wir können also, wie in der zweiten Tabelle, vereinfachen. Das Sternchen bedeutet ‚Egal'. Falls also die erste Frage verneint wird, braucht die zweite Frage nicht mehr gestellt zu werden.

Beispiel/ Frage	Gutes Wetter	Genug Zeit?	Entscheidung
1.	JA	JA	HINGEHEN
2.	JA	NEIN	NICHT
3.	NEIN	*	NICHT
4.	NEIN	*	NICHT

Genauso füttert man auch Expert-Ease mit Informationen. Zuerst wird ein Gerüst der Tabelle erstellt, in der Fragen und denkbare Entscheidungen stehen (das Ergebnis-Fenster). Diese Tabelle wird dann mit den möglichen Antworten aufgefüllt (Beispiel-Fenster). Bei Expert-Ease muß man allerdings nicht nur mit ‚ja' oder ‚nein' antworten – es stehen viele Möglichkeiten offen. Sie werden zum ‚Ankreuzen' in das Beispiel-Fenster eingetragen. Ist das Beispiel-Fenster fertig, dann leitet Expert-Ease mit ACLS Regeln aus der Tabelle ab, mit denen es später arbeitet. Auch auf Widersprüche macht ACLS uns aufmerksam, etwa durch: „11 Regeln abgeleitet, Beispiele 1, 5, 6 widersprechen den Regeln".

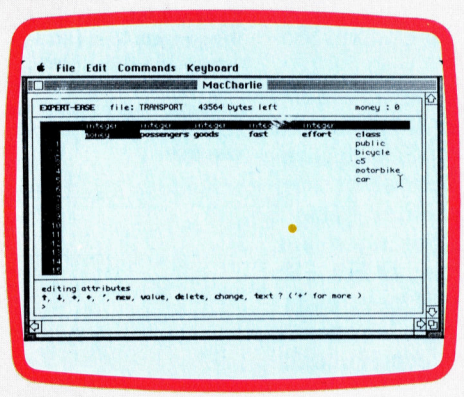
Expert-Ease ist zweifellos ein außerordentlich leistungsfähiges und durchdachtes Programm. In vielen Fällen ist es den kommerziellen Systemen durchaus gewachsen. Ein weiteres Plus ist die Fähigkeit zur Verkettung einzelner Modelle, um komplexe Probleme in handliche Stücke zu zerlegen. Der wesentliche Unterschied zwischen einem Expert-Ease-Modell und einem kommerziellen Expertensystem besteht darin, daß Expert-Ease keine Informationen darüber gibt, wie es zu seinen Entscheidungen kommt. Man kann zwar Nachrichten in den Text der Entscheidung einfügen, der Weg der Entscheidungsfindung bleibt aber verborgen. Das kann außerhalb der professionellen Nutzung auch ein Vorteil sein – die Grundlagen einer Entscheidung sind oft sehr komplex. Fachleute müssen jedoch oft ein Glied oder gar die ganze Argumentationskette Schritt für Schritt prüfen.

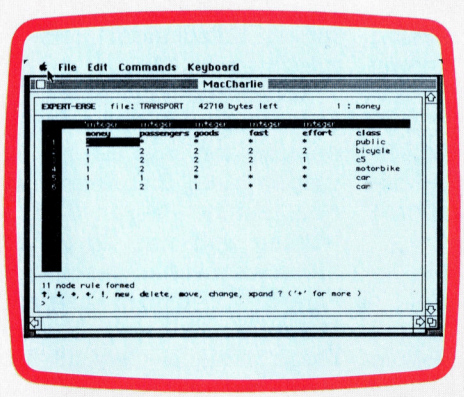
Anwendungsbereiche

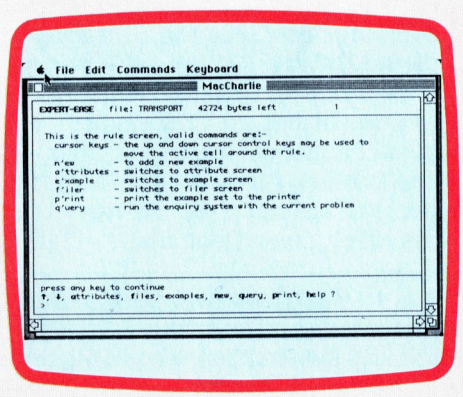
Expert-Ease kann seine Fähigkeiten besonders bei der Erzeugung einfacher Modelle hervorragend beweisen.

Ein Gebiet für den Einsatz von Expert-Ease ist die Bearbeitung von Kreditanträgen. Das Anwenderspektrum reicht dabei vom Kleinbetrieb, der sich zwischen den Zahlungsbedingungen „30 Tage Ziel", Barzahlung oder Vorkasse entscheiden muß, bis zur Großbank, die Kredite über größere Summen bewilligen soll. Jede Kreditbewilligung beinhaltet ein Risiko, das beurteilt werden muß.

Normalerweise wird die Kreditentscheidung von einem Bankangestellten getroffen. Wenn







Ein Modell wird entwickelt

Der erste Schritt bei der Entwicklung eines Expert-Ease Modells ist der Entwurf eines ‚Skeletts', auf dem die Entscheidungstabelle aufbaut. Bei Expert-Ease wird sie als ‚Ergebnis-Fenster' bezeichnet und durch die Taste „A" aufgerufen. Um dieses Fenster zu füllen, müssen die notwendigen Fragen und ihre Reihenfolge festgelegt werden. Jede Frage erhält einen Namen, der als Spaltenüberschrift ausgegeben wird.

Als nächstes sind die möglichen Entscheidungen (oder Empfehlungen) festzulegen. Sie erhalten gleichfalls Namen, die rechts im Fenster als Zeilenbezeichnungen erscheinen. Bei einem „Transportmodell" könnten wir uns für die Fragen nach Kosten, Anzahl der Passagiere, Gepäck, Geschwindigkeit und Bequemlichkeit entscheiden.

Nach den Überschriften können mit „T" die Texte für Fragen und Empfehlungen eingegeben werden.

Bei der Textlänge gibt es keine Beschränkungen – was nicht mehr ins Fenster paßt, wird automatisch in ein zweites geschoben.

Das Beispielfenster ist die eigentliche Entscheidungstabelle. Es heißt so, weil Expert-Ease an Beispielen lernt, welche Entscheidungen zu fällen sind. Im ersten Beispiel antwortet ein Benutzer mit ‚2' (Nein) auf die Frage nach Geld, das heißt, er will es möglichst billig haben. Die Antworten auf weitere Fragen werden damit irrelevant, und wir füllen die Zeile mit Sternchen, den Zeichen für ‚egal'. Expert-

ein Expert-Ease Modell ihn entlasten soll, muß zuerst ein Kriterienkatalog der Eigenschaft „kreditwürdig" erstellt werden, und die dazugehörigen Fragen sind zu formulieren. Aus den Antworten läßt sich das Risiko abschätzen. Ein vereinfachtes Modell für Kreditvergaben haben wir in Tabellenform gefaßt.

Ein typischer Programmablauf wäre dann:

Sicherheiten	Schulden	Überzogen	Regelm. Einkommen	Kredit
JA	*	*	*	JA
NEIN	JA	*	*	NEIN
NEIN	NEIN	JA	JA	WEITERLEITEN
NEIN	NEIN	JA	NEIN	NEIN
NEIN	NEIN	NEIN	NEIN	WEITERLEITEN
NEIN	NEIN	NEIN	JA	JA

Bietet der Kunde Sicherheiten? Nein
War der Kunde bereits im Verzug? Nein
Hat der Kunde sein Konto je über Limit belastet? Ja
Hat der Kunde ein regelmäßiges, belegtes Einkommen? Ja
Bitte Antrag zur Entscheidung an den Vorgesetzten weiterleiten

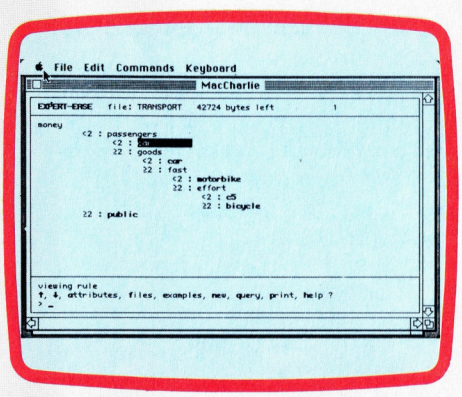
Anpassungsfähig

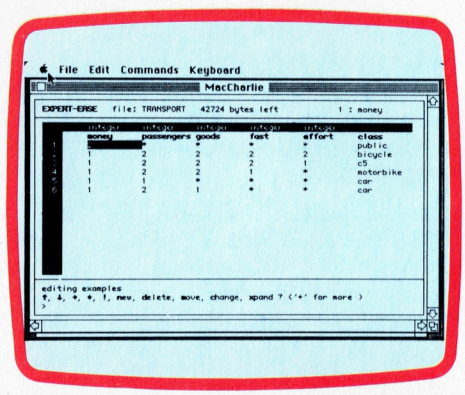
Auf diese Art können einfache Entscheidungen mit Hilfe des Expertensystems auch von weniger qualifiziertem Personal getroffen werden. Nur kompliziertere Fälle müssen auf höherer Ebene entschieden werden. Sollte die Bank ihre Kreditbedingungen ändern, also etwa bei allen Krediten über DM 5000,– Sicherheiten fordern, dann läßt sich das Modell problemlos den neuen Gegebenheiten anpassen. Die Umprogrammierung ist einfach.

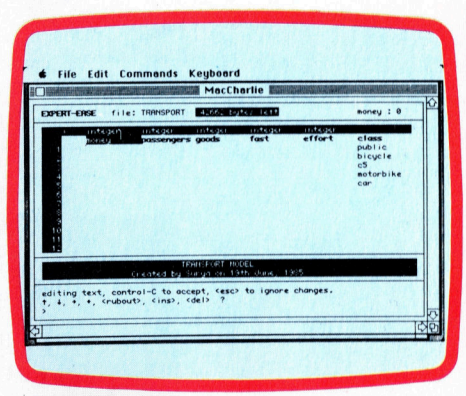
Expertensysteme sind eine ideale Lösung bei diagnostischen Problemen. Komplexe Probleme, etwa in der Medizin, erfordern große Datenmengen mit komplexen Regeln und feinen Unterscheidungen. Hier muß ein maßgeschneidertes System benutzt werden.

Eine kleine Fluggesellschaft etwa, die per Helikopter das Personal auf Ölbohrplattformen transportiert, kann Expert-Ease für die Entscheidungsfindung bei der Wartung ihrer Hubschrauber-Flotte einsetzen. Bei kleinen Reparaturen wird normalerweise ein Ingenieur gebraucht, der den Hubschrauber inspiziert und anordnet, welche Arbeiten die geringer qualifizierten Mechaniker ausführen sollen. Mit Hilfe von Expert-Ease dagegen kann der Mechaniker selbst herausfinden, welche Wartungsarbeiten fällig sind.

Dieses Verfahren läßt sich bei Reparatur und Wartung jedes mechanischen, elektrischen oder elektronischen Gerätes anwenden – nicht zuletzt auch bei Computern. Tatsächlich wird Expert-Ease sowohl bei der Untersuchung von Softwarefehlern als auch bei Problemen mit der Hardware eingesetzt. Hat ein Kunde Schwierigkeiten, dann müssen im Regelfall geschulte Hard- und Software-Spezialisten kostbare Zeit aufwenden, um den Fehler zu lokalisieren. Mit Expert-Ease ist es dagegen möglich, diese Aufgaben an weniger geschultes Personal zu delegieren. Experten müssen sich dann nur noch um die wirklich komplizierten Fälle kümmern.







Ease überspringt dann diese Fragen.

Im zweiten Beispiel muß jede Frage wirklich gestellt werden. Erst wenn wir alle Antworten kennen, kann entschieden werden, ob dem Benutzer ein Fahrrad oder ein Mofa empfohlen werden soll. Beispiel 3 zeigt die Antworten, die zur Empfehlung eines Mofas führen.

Nach Fertigstellung der Beispiele geben wir ein Ausrufungszeichen „!" ein. Expert-Ease leitet dann die ihnen zugrundeliegende Logik ab. Dazu werden alle Beispiele durchlaufen und geprüft, ob Widersprüche auftauchen (etwa unterschiedliche Antworten mit der gleichen Folgerung). Danach wird eine „Regel" (Flußdiagramm) formuliert. Wenn Widersprüche auftreten, wird die Regel zwar formuliert, aber darauf hingewiesen, daß die Beispiele x,y,z ihr widersprechen. Konflikte muß der Anwender dann selbst beseitigen.

Sobald eine Regel abgeleitet worden ist (mit oder ohne Widersprüche), kann sie durch die Taste „R" auf dem Bildschirm dargestellt werden. Man sieht, daß die erste Frage unter der Überschrift „Geld" gespeichert ist, und daß auf einen Wert unter 2 (Ja) die „Fahrgast"-Frage folgt, während 2 oder größer zur Empfehlung ‚Öffentlich' führt. Widersprüche lassen sich am besten mit dieser Tabelle korrigieren.

Das System fragt, zu welchem Befehl Hilfe gewünscht wird. Dabei liefert „H" eine Zusammenfassung aller in diesem Fenster gültigen Befehle. Gibt man den Anfangsbuchstaben eines anderen Befehls ein, erhält man eine ausführlichere Beschreibung.

Fairer Tausch

Sie haben nun sicher die Neue Welt erreicht und können mit dem Handel beginnen. Mit ein wenig Geschick lassen sich indianische Schnitzereien, Gewürze und Perlen eintauschen, die Ihnen nach Ihrer Rückkehr einen guten Gewinn einbringen.

Die Handels-Routine wird in Zeile 892 des Hauptprogramms aufgerufen. Zuerst wird überprüft, ob Sie Waffen mitgebracht haben. Der Häuptling der Eingeborenen hat eine gesunde Abneigung gegen Waffen und lehnt Ihr Waffenangebot dankend ab. Mit den übrigen Waren kann gehandelt werden.

Sie werden nun informiert, welchen Wert die vom Häuptling angebotene Ware besitzt. Zuerst bieten Sie Salz. Sie können dafür Schnitzereien, Gewürze oder Perlen erhalten. Versuchen Sie Ihr Glück dann mit Stoffen, Messern und Juwelen.

Waffenhandel

Wenn Sie keine Waffen an Bord haben, braucht der Häuptling Ihnen auch nicht mitzuteilen, daß er an einem Handel kein Interesse hat. Daher wird in Zeile 10072 das zweite Element des Arrays OA überprüft, in dem die Anzahl der Waffen verzeichnet ist. Ist dieser Wert 0, wird die Meldung des Häuptlings selbstverständlich nicht ausgegeben.

Zeile 10080 überprüft die letzten vier Elemente des Arrays OA. Die ersten beiden Elemente repräsentieren Medizin und Waffen und werden daher nicht untersucht. Sind Messer, Salz, Stoffe oder Juwelen an Bord, wird Ihnen mitgeteilt, daß der Häuptling im Austausch Perlen, Schnitzereien und Gewürze anbietet. Sind alle vier Elemente des Arrays auf 0 gesetzt, endet das Spiel. Sie sind dann nämlich „handelsunfähig".

Nachdem Sie sich über den Wert der einzelnen Waren informieren konnten, wird in den Zeilen 10130 bis 10200 eine Schleife zum Gütertausch gestartet. So berechnet das Programm etwa die Mengen an Perlen, Schnitzereien und Gewürzen, die der Häuptling für eine Einheit Ihrer Waren bietet, und Sie werden gefragt, welche Waren Sie haben wollen. Anschließend speichert der Rechner die entsprechenden Werte zur Weiterverarbeitung.

Der Schleifenzähler behandelt jedes Handelsgut separat. In Zeile 10135 wird überprüft, ob das gerade bearbeitete Array-Element auf 0 gesetzt ist. Wenn ja, besitzen Sie diese Ware nicht, und durch die NEXT-Anweisung in Zeile 10200 wird mit dem nächsten Element von OA() fortgefahren. Ansonsten informiert Sie Zeile 10145 über die jeweils vorhandenen Mengen. Die Zeilen 10150 bis 10153 geben entsprechend dem aktuellen Wert von T die Warenart aus.

Nachdem der Häuptling ein Angebot unter-

breitet hat, berechnet das Programm die Mengen seiner Waren, die Sie im Tausch für Ihre Waren erhalten. Für diese Berechnungen wird das in Zeile 63 erstellte zweidimensionale Array EQ(,), das die zugehörigen Tauschwerte enthält, verwendet.

Der Ausdruck in Zeile 10165 berechnet für jede angebotene Ware die Perlenanzahl, indem die Menge der zu tauschenden Ware OA(T) mit der entsprechenden Tauschrate aus EQ(,) multipliziert wird. Da die Schleife von 3 bis 6 läuft, muß von T der Wert 2 subtrahiert werden, um eine Übereinstimmung mit der Element-Nummer in EQ(,) zu erzielen. Wird Salz gehandelt, ist der Schleifenzähler T=3. EQ(T−2,1) entspricht somit der Schnittstelle zwischen dem ersten Element des ersten Unterbereiches (Salz) und dem ersten Element des zweiten Unterbereiches (Perlen). Das ist hier der Wert 0,5. Also wird in Zeile 10165 die Salzmenge mit 0,5 multipliziert, um die Perlenanzahl für das Tauschgeschäft zu ermitteln.

In Zeile 10166 wird eine gleichartige Funktion für die Handelsmenge von Schnitzereien ausgeführt, und in Zeile 10167 kommen die Gewürze an die Reihe.

Alles an Bord

Nachdem Sie das Angebot des Häuptlings kennen, müssen Sie entscheiden, welche Waren Sie erwerben wollen. Das gewünschte Gut wird mit 1, 2 oder 3 gewählt. Anschließend müssen die ertauschten Waren auf das Schiff gebracht werden: Zeile 10180 speichert die Menge im Array AO(), das bereits zuvor in Zeile 68 dimensioniert wurde.

Die drei Elemente dieses Arrays repräsentieren die drei vom Häuptling angebotenen Waren. Die Menge der auf das Schiff gebrachten Güter wird nochmals unter Verwendung der Tauschraten aus EQ(,) und Multiplikation des entsprechenden Elementes mit der Menge der gehandelten Ware, OA(T), errechnet. I, die Nummer, die Sie zur Auswahl einer Ware eingeben müssen, wird zur Bestimmung des korrekten Elementes von AO() verwendet. In AO() werden die gehandelten Perlen, Schnitzereien und Gewürze gespeichert. Außerdem wird I zur Kennzeichnung des zweiten Unterbereiches von EQ(,) verwendet.

Nachdem die neu erworbenen Güter in AO() verzeichnet sind, kehrt die Schleife, falls notwendig, zum Anfang zurück. Wenn alle Waren ausgetauscht sind, wird der Handel abgebrochen, und es wird Ihnen die Menge der nun an Bord befindlichen Perlen, Schnitzereien und Gewürze mitgeteilt. Dazu dienen die Zeilen 10220 und 10244. Danach erfolgt der Rücksprung zum Hauptprogramm.

Im nächsten, abschließenden Artikel komplettieren wir das Spiel. Sie bekommen es mit einem Aufstand zu tun und müssen die frisch erworbenen Waren in der Heimat verkaufen.

Modul Zwölf: Der Handel

Ergänzungen am Hauptprogramm

```
892 GOSUB10070
```

Die Handels-Routine

```
10070 PRINTCHR$(147):GOSUB9200:REM TRADING
10072 IFOA(2)=0THEN10080
10074 S$="THE CHIEF DOESN'T WANT YOUR*":GOSUB9100
10076 S$="GUNS - THEY WILL CAUSE TROUBLE*":GOSUB91
00
10078 PRINT:GOSUB9200
10080 IFOA(3)<>0OROA(4)<>0OROA(5)<>0OROA(6)<>0THEN
10100
10085 S$="YOU HAVE NO GOODS LEFT*":GOSUB9100
10090 S$="FOR TRADING*":GOSUB9100
10095 GOTO10038
10100 S$="IN EXCHANGE FOR ANY KNIVES*":GOSUB9100
10102 S$="SALT CLOTH OR JEWELS YOU HAVE*":GOSUB910
0
10104 S$="HE OFFERS YOU PEARLS CARVINGS*":GOSUB910
0
10106 S$="AND SPICES*":GOSUB9100
10108 PRINT:GOSUB9200
10110 S$="WHEN YOU LEFT PORT THESE WERE*":GOSUB910
0
10112 S$="WORTH:*":GOSUB9100
10114 S$="PEARLS - 2 GLD PCS EACH*":GOSUB9100
10116 S$="CARVINGS - 2 GLD PCS EACH*":GOSUB9100
10118 S$="SPICES - 1 GLD PC PER GRAM*":GOSUB9100
10120 PRINT:GOSUB9200
10122 S$="BUT THEIR VALUE MAY HAVE*":GOSUB9100
10124 S$="CHANGED WHEN YOU GET HOME*":GOSUB9100
10125 PRINT:GOSUB9200:S$=K$:GOSUB9100
10126 GETI$:IFI$=""THEN10126
10130 FORT=3TO6
10135 IFOA(T)=0THEN10200
10140 PRINTCHR$(147):GOSUB9200
10145 PRINT"YOU HAVE";OA(T);
10150 IFT=3THENS$="BAGS OF SALT*"
10151 IFT=4THENS$="BALES OF CLOTH*"
10152 IFT=5THENS$="KNIVES*"
10153 IFT=6THENS$="JEWELS*"
10155 GOSUB9100
10156 PRINT:GOSUB9200
10160 S$="FOR THESE THE CHIEF OFFERS YOU*":GOSUB91
00
10165 PRINT"EITHER";OA(T)*EQ(T-2,1);"PEARLS"
10166 PRINT"    OR";OA(T)*EQ(T-2,2);"CARVINGS"
10167 PRINT"    OR";OA(T)*EQ(T-2,3);"GRAMS OF SPIC
E"
10168 PRINT:GOSUB9200
10170 S$="DO YOU WANT PEARLS,CARVINGS*":GOSUB9100
10172 S$="OR SPICES?*":GOSUB9100
10174 S$="(ENTER 1,2 OR 3)*":GOSUB9100
10175 INPUTI$
10176 I=VAL(I$):IFI<1ORI>3THEN10174
10180 AO(I)=AO(I)+(OA(T)*EQ(T-2,I))
10190 PRINT:PRINT"THE ";T$(I);" ARE PUT ON THE SHI
P"
10192 S$=K$:GOSUB9100
10194 GETI$:IFI$=""THEN10194
10200 NEXT
10210 PRINT:PRINT:GOSUB9200
10215 S$="    END OF TRADING*":GOSUB9100
10216 PRINT:GOSUB9200
10218 S$="YOU HAVE OBTAINED:*":GOSUB9100
10220 PRINTAO(1);"PEARLS"
10222 PRINTAO(2);"CARVINGS"
10224 PRINTAO(3);"GRAMS OF SPICE"
10226 PRINT:GOSUB9200
10228 S$=K$:GOSUB9100
10229 GETI$:IFI$=""THEN10229
10230 RETURN
```

BASIC-Dialekte

Spectrum:
Ersetzen Sie im gesamten Programm EQ(,) durch Q(,) und AO() durch E() und ändern Sie:

```
10070 CLS
10126 LET I$=INKEY$:IF I$="" THEN GO TO 10126
10140 CLS:GO SUB 9200
10229 LET I$=INKEY$:IF I$="" THEN GO TO 10299
```

Acorn B:
Ändern Sie das Programm wie folgt:

```
10070 CLS
10126 I$=GET$
10140 CLS:GOSUB 9200
10229 I$=GET$
```

Gestatten, Dr. LOGO

LOGO ist mit eine der besten Einstiegssprachen in die Welt der Programmierung. Wir sehen uns Dr. LOGO für den Schneider an und ziehen Vergleiche zu der ursprünglichen Version für den IBM PC.

Als LOGO von Großcomputern auf Acht-Bit-Microcomputer übertragen wurde, waren Einschränkungen und Vereinfachungen nötig, um die Sprache auf Geräten mit lediglich 64 KByte überhaupt lauffähig zu machen. Kurz nach Erscheinen der ersten 16-Bit-Micros gab es mehrere erweiterte Versionen.

Gary Kildall – der Gründer von Digital Research (Hersteller des Betriebssystems CP/M) – war von dem Potential der Sprache fasziniert. Mit Dr. LOGO entwickelte er eine eigene Version für den IBM PC, die es auch für eine Reihe weiterer 16-Bit-Maschinen gibt.

Inzwischen ist Dr. LOGO auch für Acht-Bit-CP/M-Maschinen erhältlich. Das Programmpaket des Schneider CPC 464 enthält beispielsweise eine Version.

Das ursprüngliche Dr. LOGO für den IBM PC benötigt mindestens 192 KByte RAM. Die Programmdiskette ist kopiergeschützt, wird jedoch gleich mit Sicherheitskopie geliefert. Nach Einlegen der Diskette lädt ein Systemreset automatisch das Sprachsystem. Die Auflösung der Farbgrafik beträgt 320 × 200 Pixel. Es stehen 16 Hintergrundfarben und vier Sets zu je drei Pen (Vordergrund) Farben dem Anwender zur Verfügung.

Im Handbuch ist Dr. LOGO als Erweiterung des Apple LOGO beschrieben. Es enthält alle LOGO-Befehle der Apple Systeme, darunter die Standardbefehle für Grafik, Listenverarbeitung und Speicherverwaltung, wie auch „Primitives" (LOGO-Grundbefehle bzw. -routinen) für Fehlerbehandlung, „Properties" (eine Methode, einem Objekt mehr als einen Wert zuzuweisen) und „Packages" (eine Technik, mit der sich Prozeduren zu Gruppen zusammenfassen lassen).

Der Editor arbeitet mit den Funktionstasten des IBM, erkennt aber auch die in Apple-LOGO gebräuchlichen Steuercodes. Wird während des Programmablaufs ein Fehler gefunden, so kann man mit der Eingabe „ed" den Editor aufrufen, der die fehlerhafte Prozedur gleich zur Korrektur bereithält. Leider gibt der Bildschirm keine Auskunft darüber, ob der Editiermodus eingeschaltet ist. Die meisten anderen LOGO-Versionen zeigen dies mit einer speziellen Statuszeile am unteren Bildschirmrand an.

LOGO Benchmarks

PROZEDUR		ERGEBNIS			
		IBM PC (256K)	SCHNEIDER CPC 464/664	COMMODORE 64	APRICOT F1e
RECUR	erreichte Ebene	512	187	90	832
DEFVARS		1307	263	553	1606
ENDRECUR1		512	189	INFINITE	INFINITE
ENDRECUR2		151	129	65	72
LINES (Mit Turtle)	Zeit (in Sekunden)	9	53	13	10
LINES (Ohne Turtle)		5	39	11	6
ASSIGN		25	22	6	20
LISTS		64	74	18	69
ARITH		80	138	36	77
RECURA		92	91	24	72

Befehlsvielfalt

Dr. LOGO kennt mehr als die üblichen Primitives zur Listenverarbeitung:

sort sortiert eine Liste in alphabetischer Reihenfolge

shuffle stellt die Elemente einer Liste in zufällige Reihenfolge

piece gibt Ihnen die Möglichkeit, einen Teil einer Liste auszuwählen

Dr. LOGO besitzt außerdem eine Reihe von Fehlersuchroutinen:

watch zeigt den Arbeitsvorgang einer Prozedur zeilenweise an

trace gibt die Definition der Variablennamen in der Reihenfolge ihres Aufrufs aus

debug teilt den Bildschirm in zwei Fenster. Im „debug"-Fenster lassen sich die Informationen von trace und watch darstellen, während das „Programm"-Fenster den Programmablauf anzeigt.

Die Prozeduren können mit Bemerkungen ver-

Die Benchmark-Prozeduren

Die folgenden neun Benchmark-Prozeduren testen unterschiedliche Eigenschaften von LOGO und zeigen die Stärken und Schwächen der einzelnen Sprachversionen.

- RECUR mißt die Größe des Stacks.
- DEFVARS mißt den Platz für die Definition von Variablen und Prozeduren.
- ENDRECUR1/ENDRECUR2 bestimmen, ob die Endrekursion korrekt implementiert wurde. Eine ideale Implementation sollte unendliche Rekursionen zulassen. Während ENDRECUR2 als Ablauf das Ergebnis ausgibt, ist ENDRECUR1 ein Befehl.
- LINES mißt die Zeichengeschwindigkeit. Zahlenpaare zeigen an, wieviel Zeit gewonnen wird, wenn die Turtle beim Zeichnen nicht auf dem Bildschirm erscheint.
- ASSIGN mißt die Geschwindigkeit, mit der Variablen Werte zugeordnet werden.
- LISTS mißt die Geschwindigkeit der Operatoren für Listenverarbeitung.
- ARITH mißt die Geschwindigkeit arithmetischer Funktionen,
- RECURA das Tempo rekursiver Aufrufe.

sehen werden, die sich mit „noformat" herausnehmen lassen, falls der Platz knapp wird.

Es gibt auch eine Reihe von neuen Primitives zur Prozedurverwaltung:

follows definiert die Reihenfolge, in der die Prozeduren auf dem Bildschirm dargestellt werden

potl zeigt die Namen aller Prozeduren, die nicht schon von anderen Prozeduren aufgerufen werden

poref<name> stellt die Namen aller Prozeduren auf dem Bildschirm dar, die die Prozedur <name> ansprechen

pocall<name> zeigt die Namen aller Prozeduren, die von der Prozedur <name> aufgerufen werden

All diese Zusatzmodule sind bei der Entwicklung von LOGO-Programmen eine große Hilfe.

Aber Dr. LOGO hat auch eine Schwäche: Der Editor besitzt keine Funktion fürs Suchen und Tauschen, es gibt keine Schnittstelle zum Maschinencode und auch keine Dateiverarbeitung. Das 300 Seiten starke Handbuch hingegen ist zu loben. Außer einer LOGO-Einführung enthält es einen Überblick, der jedem Primitive eine eigene Seite widmet.

Interessant ist ein Vergleich zwischen Dr. LOGO und den Acht-Bit-Versionen der Sprache. Der Arbeitsbereich von LOGO wird in „Nodes" (Datenelementen) gemessen. Beim Systemstart zeigt Dr. LOGO die beruhigende Meldung, daß rund 10 000 Nodes zur Verfügung stehen (Acht-Bit-Systeme haben normalerweise 2–3000). Es gibt also genug Platz für die Definition von Prozeduren und den Ablauf rekursiver Prozeduren. Die Endrekursion (eine rekursive Prozedur mit dem rekursiven Aufruf in der letzten Zeile) wurde jedoch schlecht implementiert. Programme, die auf dem Commodore oder Acorn B „ewig" laufen, erreichen unter Dr. LOGO schon nach einigen hundert Rekursionen die Speichergrenze.

Grafik wird schnell aufgebaut, doch sind die Listenverarbeitung wie auch arithmetische Abläufe weit langsamer als in anderen Sprachversionen gemeinhin üblich.

Dr. LOGO Musicus

Dr. LOGO gehört zum Lieferumfang der Schneider Diskettenstation. Das Sprachsystem wurde der Schneider Hardware angepaßt. So arbeitet die Grafik zwar mit dem gleichen „Pen"-Farbensystem wie das Dr. LOGO des IBM PC, doch lassen sich die Pen-Farben in einer Liste festlegen, die den Anteil von Rot, Grün und Blau bestimmen. Weiterhin fehlt der IBM-Version die große Palette der Schneider-Klangbefehle. Die Schneider-Version bietet „env" (Hüllkurve für die Lautstärke), „ent" (Hüllkurve für Tonhöhe) und „release" (Freigabe der abgeschalteten Tonkanäle).

Die Schneider-Version enthält die meisten Fähigkeiten des Apple-LOGO, darunter „Properties" und Fehlerroutinen, nicht aber die Verknüpfung von Prozeduren (Packages).

Ihr fehlen weiterhin die Fehlersuchprimitives und die Prozedurverwaltung der IBM-Version, die vermutlich der geringeren Speicherkapazität der Schneidergeräte zum Opfer fielen. Vermißt wird auch die „define"-Funktion, und Dateien lassen sich nur von CP/M aus löschen.

Da der Editor sehr langsam reagiert, können bei schneller Eingabe Zeichen verschwinden. Es ist jedoch sehr praktsich, daß der Editiercursor beim Prozedurablauf automatisch auf Fehler positioniert wird.

Mit „.examine" und „.deposit" (PEEK und POKE) bietet Schneider – im Gegensatz zu IBM – begrenzten Zugang zu Maschinencode.

Die mitgelieferte Information ist allerdings dürftig und bezieht sich auf eine LOGO-Einführung und ein Referenzhandbuch. In der 25seitigen Dokumentation fehlen einige Optionen von Dr. LOGO, die dennoch funktionieren.

Beim Systemstart zeigt Dr. LOGO, daß 2105 Nodes zur Verfügung stehen – eine auf Acht-Bit-Geräten durchaus übliche Zahl. Es wurde jedoch die Endrekursion nicht korrekt implementiert, und die Sprache ist bei der Verarbeitung von Grafik, Arithmetik und Listen etwa dreimal so langsam wie LOGO auf dem Commodore 64.

Alles in allem ist die Schneider-Version nicht ganz so interessant wie Dr. LOGO auf dem IBM PC, doch steht damit eine gute Standardversion von LOGO zur Verfügung, die mit Packaging und Fehlerroutinen erfreulich ordentlich ausgerüstet ist.

Geschäftssprache

Wir beenden unsere COBOL-Serie und versuchen herauszufinden, warum COBOL immer noch die erfolgreichste Sprache für den kommerziellen Einsatz ist.

„Und hiermit neigt sich nun der COBOL-Kurs seinem Ende zu"

Die Unterschiede zwischen Tabellen und Arrays sind auf den ersten Blick nicht leicht zu erkennen. In Arrays werden die Elemente über eine Elementnummer oder einen Index angesprochen, bei Tabellen jedoch über einen Schlüssel, der Teil jedes Datenelementes ist. Weiterhin haben Tabellenelemente normalerweise eine Datensatzstruktur und bestehen aus mehreren Feldern.

COBOL hat spezielle Fähigkeiten zur Tabellenbearbeitung, die sich auch für einfache Arrays einsetzen lassen. Dabei wird jedes Datenelement (außer den Ebenen 01 und 77) mit der Bedingung „OCCURS nn TIMES" als Wiederholung des ersten Elementes definiert (nn kann jede positive Ganzzahl sein; TIMES darf weggelassen werden). Hier die Definition eines einfachen Arrays mit 20 Ganzzahlen:

```
01 Einfach-Array.
02 Array-Element PIC 9 (5) OCCURS 20
   TIMES.
```

Die Elemente dieses Arrays werden mit „Array-Element(5)" oder „Array-Element(Zahl−1)" angesprochen, wobei Zahl−1 ein positives Ganzzahlelement darstellt. Da jedes Array einen Namen hat, können Anweisungen wie MOVE ein ganzes Array bearbeiten.

Durch die weitere Unterteilung der wiederholten Felder werden zweidimensionale (oder größere Arrays möglich:

```
01 Zwei-dimensionales-Array.
  02 Array-Zeile OCCURS 20 TIMES.
    03 Array-Element OCCURS 30 TIMES.
```

Die Elemente dieses Arrays werden mit „Array-Element(3,4)" oder „Array-Element (Zahl-1,Zahl-2)" bezeichnet.

In COBOL lassen sich die unterschiedlichsten Datenelemente zu zusammenhängenden Strukturen kombinieren. Das folgende Beispiel definiert die Preie eines Schuhgeschäftes.

```
01 Lager-Tabellen.
  02 Groesse-Beschreibung OCCURS 20
     TIMES.
    03 Groesse-englisch PIC 9V9.
    03 Groesse-metrisch PIC 99V99.
  02 Lagermenge PIC 999.
  02 Lager-Artikel-gesamt.
    03 Lager-Artikel OCCURS 500 TIMES,
      ASCENDING KEY IS Lager-Nummer.
      04 Lager-Nummer PIC X(6).
      04 Lager-Beschreibung PIC X(20).
      04 Lager-Preise PIC 999V99
        OCCURS 20 TIMES.
      04 Lager-Indikator PIC X.
      88 Lager-Wert VALUE 'Y'.
```

Sie sehen, daß alle zusammenhängenden Informationen in einer großen Tabelle zusammengefaßt sind. Die Wiederholung eines

Gruppenelementes wie Lager-Artikel wiederholt automatisch auch die darin enthaltenen Subfelder. Sie können sich daher auf Lager-Nummer(6) beziehen oder auf Lager-Preise(100,3). Da sich auch die Bedingungen der Ebene 88 mit Subscripts versehen lassen, können Abfragen, beispielsweise mit IF Lager-Wert(120), durchgeführt werden. Die Bedingung ASCENDING (oder DESCENDING) KEY ist wahlfrei und wird nur eingesetzt, wenn die Tabelle eine bestimmte Reihenfolge halten und das Verb SEARCH ALL eingesetzt werden soll. Da COBOL seine Datenelemente nicht automatisch in einer bestimmten Reihenfolge ablegt, müssen Sie sich darum kümmern.

INDEX-Datentyp

Tabellen lassen sich über jedes numerische Datenelement, das eine positive Ganzzahl enthält, mit einem Index versehen. COBOL bietet weiterhin einen speziellen INDEX-Datentyp, der mit „INDEXED BY Index-Name" an jede OCCURS-Bedingung angefügt werden kann. Dabei wird ein numerisches Datenelement definiert, das nur einen positiven Ganzzahlwert enthalten kann und das zugehörige Array indiziert. Eine Tabelle kann mehrere Indizes haben. Es ist möglich, ein beliebiges numerisches Datenelement als „USAGE INDEX" zu deklarieren, der dann zwar nicht als Index für das Array dient, aber in der Indexarithmetik verarbeitet werden kann.

Indexelemente dürfen nicht in normalen arithmetischen Anweisungen erscheinen, haben aber ein eigenes arithmetisches Verb – mit folgendem Format:

```
SET Indexname TO Numerischer-Wert.
SET Indexname UP BY Numerischer-Wert.
SET Indexname DOWN BY Numerischer-Wert.
```

Numerischer-Wert kann eine Konstante oder jedes normale numerische Datenelement sein.

Das Durchsuchen von Tabellen ist mit Abstand der wichtigste Arbeitsgang bei dieser Datenstruktur. Da der Zugriff durch Schlüsselwerte gesteuert wird, muß oft die gesamte Datei durchsucht werden, ehe ein Eintrag für einen bestimmten Schlüssel gefunden ist. COBOL unterstützt diesen Ablauf mit dem Verb SEARCH, das zwei Formate haben kann:

```
SEARCH Tabellen-Name VARYING Index-
    Name (oder Numerisches-Element).
    AT END Direkte-Anweisung.
    WHEN Bedingung-1 Direkte-Anweisung.
```

Die Bedingungen VARYING und AT END sind optional. Außerdem können beliebig viele WHEN-Bedingungen angeführt werden. SEARCH veranlaßt ein lineares Durchsuchen der Tabelle (das heißt, die Datenelemente werden nacheinander von Anfang bis Ende angesprochen). Wenn die angegebenen Bedingungen eingetreten sind und die Suche beendet ist, können bestimmte Abläufe ausgelöst werden. Direkte Anweisungen sind feste Vorgänge, die keine Wahl- oder Verzweigungsmöglichkeit haben. IF ist keine direkte Anweisung, wohl aber MOVE.

Das zweite Format von SEARCH sieht so aus:

```
SEARCH ALL Tabellen-Name.
AT END Direkte-Anweisung.
WHEN Bedingung-1 Direkte-Anweisung.
```

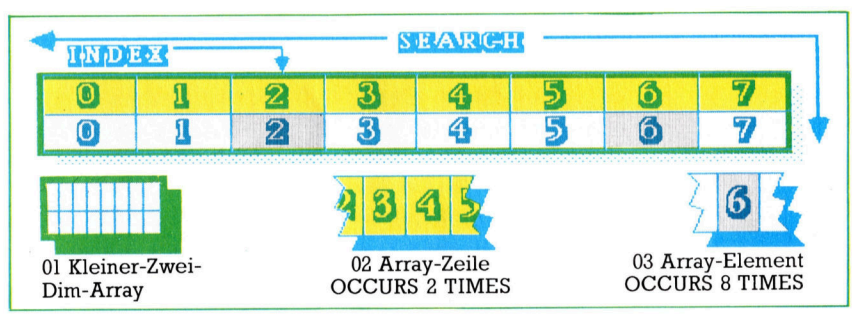
In diesem Fall ist nur eine WHEN-Bedingung möglich, die nach dem Schema

```
Daten-Element = Wert
```

(oder mehreren dieser Anweisungen, die durch ANDs verbunden sind) getestet werden muß. Im Gegensatz zur ersten Version führt dieses SEARCH-Format eine binäre Suche durch. SEARCH halbiert dabei wiederholt die Tabelle und entscheidet, in welcher Hälfte weitergesucht werden soll. Dabei müssen jedoch die Tabellenwerte in der richtigen Reihenfolge stehen, oder ASCENDING oder DESCENDING KEY in der Tabellendefinition angegeben sein.

Die Dateien externer Geräte werden im Maschinenteil definiert. Jede Datei erhält dabei mit SELECT einen Namen. Das Format:

COBOL läßt in seinem Datenteil die Definition komplexer Datensätze zu, auf die mit SEARCH und INDEX zugegriffen werden kann. Da die Arrays und ihre Zeilen separat deklariert werden, lassen sich die Daten mit Befehlen wie MOVE blockweise ansprechen und bearbeiten.




```
SELECT Datei-Name ASSIGN TO Geraete-
    Name
```

Datei-Name ist eine COBOL-Bezeichnung; Geraete-Name ist systemabhängig und kann DSK, LPT oder einen Dateinamen des Systems annehmen.

Mit Zusatzoptionen wie ORGANISATION und ACCESS läßt sich festlegen, wie die Datei auf dem Speichermedium untergebracht wird. ORGANISATION hat drei Varianten: SEQUENTIAL (vom System vorgegeben), RELATIVE (eine Datei, die über Datensatznummern für den direkten Zugriff vorbereitet wurde) und INDEXED. Bei der Dateiorganisation SEQUENTIAL ist nur der ACCESS Typ SEQUENTIAL möglich. Die anderen beiden Varianten von ORGANISATION erlauben jedoch auch den Zugriff mit RANDOM (direkt auf einen bestimmten Datensatz) oder DYNAMIC – einer Kombination von RANDOM und SEQUENTIAL.

Bei Dateien, die mit INDEXED organisiert sind, muß ein RECORD KEY (Schlüsselfeld) deklariert werden. Schlüsselfeld kann jedes Datensatzfeld sein, das von einem Index angesprochen wird. Bei Dateien vom Typ RELATIVE

muß ein RELATIVE KEY deklariert werden, der mit einem numerischen Wert die Nummer des Datensatzes bezeichnet.

Jeder in SELECT aufgeführt Dateiname muß in der FILE SECTION des Datenteils erscheinen. Neben Systeminformationen wie Buffergröße taucht hier der Dateiname in einer FD-Deklaration (File Definition) auf. Ihm folgen die Felddefinitionen der Datensätze.

Der Verarbeitungsteil enthält eine Reihe von Verben, die speziell für die Dateiverarbeitung gelten. Jede Datei muß vor dem Zugriff eröffnet werden:

```
OPEN Datei-Name FOR Zugriffs-Art.
```

Zugriffs-Art kann INPUT, OUTPUT oder INPUT-OUTPUT sein. Nach Ende der Bearbeitung wird die Datei mit

```
CLOSE Datei-Name.
```

wieder geschlossen. Das Lesen und Schreiben geschieht mit den Verben READ und WRITE, die wie alle COBOL-Verben viele Optionen besitzen, darunter Datensatz- und Dateisperren für den Mehrplatzbetrieb. Das Grundformat:

```
READ Datei-Name.
WRITE Datensatz-Name.
```

READ benötigt den Dateinamen und die Ausführung eines Befehls, der den (im Datenteil) für diese Datei definierten Datensatz füllt, WRITE dagegen nur den Namen des Datensatzes, dessen Inhalt dann an das per SELECT definierte Gerät übermittelt wird.

Beim sequentiellen Dateizugriff liest READ den nächsten Datensatz aus der Datei und rückt dann auf den darauffolgenden Satz vor. Ist das Dateiende erreicht, wird erst die Endmarkierung gelesen und dann geprüft, ob sie das Dateiende bezeichnet. READ muß die Bedingung „AT END" enthalten. AT END gibt an, was beim Erreichen der Endmarkierung geschehen soll. Normalerweise wird in diesem Fall ein Datenelement als Flag verwandt:

```
77 e-o-f PIC X VALUE 'N'.
88 End-Markierung VALUE 'Y'.
 . . .
PROCEDURE DIVISION.
MAIN-CONTROL-PARAGRAPHS.
   OPEN INPUT In-Datei, OUTPUT Out-
      Datei.
   READ In-Datei AT END MOVE 'Y' TO e-o-f.
   PERFORM Prozess-Datensatz-Paragraph
      UNTIL End-Markierung.
PERFORM Datei-Schließen.
      STOP RUN.
PROCESS-RECORD-PARAGRAPH.
   . . .
WRITE Out-Datei.
   READ In-Datei AT END MOVE 'Y' TO e-o-f-
```

WRITE schreibt die neuen Datensätze an das jeweilige Dateiende.

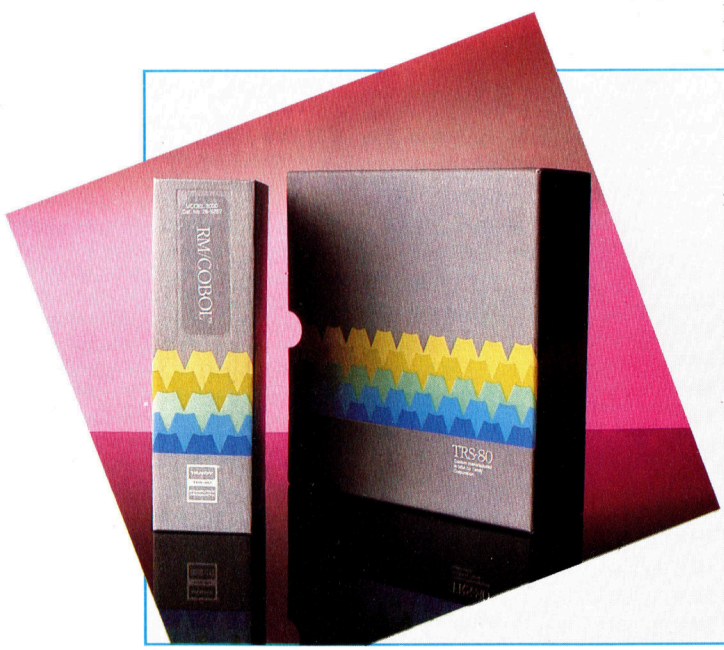
Beim Dateizugriff mit RANDOM oder DYNAMIC und der Dateiorganisation INDEXED oder RELATIVE läuft das Lesen und Schreiben in zwei Stufen ab. Zunächst wird ein entsprechender Wert im Schlüsselfeld abgelegt und – im Fall von RELATIVE – die Dateisatznummer im RELATIVE KEY. Dann erst kann der Befehl READ ausgegeben werden.

Dateien im Format INDEXED müssen den entsprechenden Wert in dem Feld RECORD KEY vorfinden. Wenn es zu dem Schlüssel keinen Datensatz gibt, wird hier statt AT END die Bedingung INVALID KEY ausgeführt.

In diesen beiden Fällen schreibt WRITE nur die neuen Datensätze. Bereits bestehende Datensätze werden mit REWRITE aktualisiert oder mit DELETE gelöscht. REWRITE und DELETE brauchen ebenfalls die Bedingung INVALID KEY.

```
READ Datei-Name NEXT RECORD.
```

kann in jedem Fall sequentiell auf eine Datei zugreifen.



Die COBOL-Reihe

Da COBOL im Geschäftsbereich weit verbreitet ist, gibt es zwar viele Anwendungen für Microcomputer, aber nur wenige für Heimgeräte (die Speicherkapazität reicht hier selten aus). Ein COBOL-System muß mit mindestens einem Diskettenlaufwerk ausgestattet sein. Einige der bekannteren Versionen sind CIS-COBOL von Microfocus, Microsoft COBOL und RM/COBOL, die unter CP/M und MS-DOS laufen.

Die Pakete lassen sich zwar auf Heimcomputern mit CP/M einsetzen (wenn genug Speicher vorhanden ist), sie sind jedoch recht teuer. Nevada COBOL bietet eine preisgünstige Alternative. Es kann bei NewStar Software, 45 Plovers Mead, Wyatts Green, Doddinghurst, Essex, CM15 0PS bestellt werden.

Kontakt zum CIA

In der vorigen Folge wurden die E/A-Möglichkeiten des C 64 behandelt. Wir wollen nun untersuchen, wie die beiden 6526 CIA Chips die Ein- und Ausgabe durchführen.

Der Commodore 64 besitzt zwei 6526 CIA- (Complex Interface Adaptor) Chips, die sich speziell um die Kommunikation mit der Außenwelt kümmern. Die CIAs sind jedoch nicht die einzigen Chips mit E/A-Funktionen – auch der 6510 und die Video Chips steuern Teilbereiche der Ein- und Ausgabe. Ein 6526-Chip besitzt zwei Acht-Bit-Datenausgänge, die beide über individuell programmierbare Leitungen verfügen. Der Chip kann Daten im Acht- oder 16-Bit-Format übertragen und hat außerdem zwei 16-Bit-Timer, die sich verbinden lassen. Er verfügt weiterhin über ein Acht-Bit-Shift-Register für die serielle Datenübertragung und eine programmierbare 24-Stunden-Uhr für vorhersehbare Ereignisse.

„Data Ready"

Der Chip hat zwei Handshake-Leitungen: PC und Flag. PC wird für einen Taktzyklus auf niedrig geschaltet, nachdem Daten zum Ausgang B des 6526 gesandt wurden, und kann daher einem externen Gerät das Signal „Data Ready" anzeigen. Die Flag-Leitung kann die Steuersignale eines externen Gerätes entgegennehmen. Damit läßt sich beispielsweise das Flag-Bit des Interrupt-Registers setzen.

Auf dem Commodore 64 steuert jeder der beiden 6526 Chips unterschiedliche E/A-Bereiche: CIA #1 (mit Basisadresse bei $DC00) ist für Tastatur und Joystick abgestellt und CIA #2 (mit Basisadresse bei $DD00) für die Datensteuerung der seriellen Schnittstelle und des User Ports. Der Video Chip führt die E/A zum Monitor aus, während der 6510 den Recorderausgang direkt steuert.

Das Programm Parawedge zeigt, wie der 6526 direkt für E/A programmiert wird. Die Routine ist ein NMI-„Keil" und arbeitet mit der Flag-Leitung. Sie sendet einen bestimmten Speicherblock als parallele Daten zum User Port oder empfängt Acht-Bit parallele Daten, bis der definierte Speicherblock voll ist. Da die Daten per NMI ausgesandt oder empfangen werden, kann die Maschine in der übrigen Zeit andere Aufgaben ausführen. Bei hohen Übertragungsraten muß sich der Commodore 64 allerdings hauptsächlich den NMI-Routinen widmen und andere Vorgänge vorübergehend „vernachlässigen".

Parawedge richtet auf dem Commodore 64 eine Acht-Bit parallele Datenübertragung mit einem externen Gerät (beispielsweise einem anderen Computer oder einem Paralleldrucker) ein, die in beide Richtungen funktioniert und über den User Port abläuft. Dabei werden die Kontakte PB0 bis PB7 für die Datenübertragung genutzt, Flag 2 für eingehende Handshakesignale, PA2 für das Signal „sendebereit" und PC2 für die Bedingung „Daten korrekt empfangen". Vor Einsatz des Programms müssen Sie zunächst festlegen, welchen RAM-Bereich Sie für die ein- oder ausgehenden Daten einsetzen wollen. Dabei werden die Anfangs- und Endadresse (im Format Lo-/Hi-Byte) mit POKE an die vier Speicherstellen von 50768 bis 50771 übergeben.

Speicherstelle 50772 gibt an, ob das Programm senden oder empfangen soll. Eine Eins schaltet auf Sendung, eine Null auf Empfang. Der eigentliche Programmcode beginnt schließlich bei SYS 50775.

Das Programm Parawedge für den Commodore 64

BASIC-Ladeprogramm

```
1000 REM ** PARAWEDGE BASIC LOADER **
1010 DATA173,84,198,208,61,169,0,141,3
1020 DATA221,169,144,141,13,221,173,2
1030 DATA221,9,4,141,2,221,173,0,221,9
1040 DATA4,141,0,221,173,80,198,133,251
1050 DATA173,81,198,133,252,173,24,3
1060 DATA141,85,198,173,25,3,141,86,198
1070 DATA120,169,188,141,24,3,169,198
1080 DATA141,25,3,88,96,169,255,141,3
1090 DATA221,169,144,141,13,221,173,24
1100 DATA3,141,85,198,173,25,3,141,86
1110 DATA198,120,169,234,141,24,3,169
1120 DATA198,141,25,3,88,96,169,144,44
1130 DATA13,221,240,36,173,1,221,145
1140 DATA251,230,251,208,2,230,252,173
1150 DATA82,198,197,251,173,83,198,229
1160 DATA252,144,49,173,0,221,41,252
1170 DATA141,0,221,9,4,141,0,221,108,85
1180 DATA198,169,144,44,13,221,240,246
1190 DATA177,251,141,1,221,230,251,208
1200 DATA2,230,252,173,82,198,197,251
1210 DATA173,83,198,229,252,144,3,108
1220 DATA85,198,120,173,85,198,141,24,3
1230 DATA173,86,198,141,25,3,88,108,24
1240 DATA3
1250 DATA25596:REM*CHECKSUM*
1260 CC=0
1270 FORI=50775TO50971
1280 READX:CC=CC+X:POKEI,X
1290 NEXT
1300 READX:IFCC<>XTHENPRINT"CHECKSUM
  ERROR" :
1310 END
```

Das Assemblerlisting von Parawedge kann eingegeben und mit einem Assembler in den Maschinencode übersetzt werden. Über das BASIC-Ladeprogramm läßt sich das Programm auch mit DATA-Befehlen einlesen.

Das Programm im Assemblercode

```
;**************************************
;*                                    *
;*       PARAWEDGE - A SEND/RECEIVE   *
;*                                    *
;*  WEDGE PROGRAM FOR 8 BIT PARALLEL  *
;*                                    *
;*    COMMUNICATIONS ON THE CBM 64    *
;*                                    *
;**************************************
;
  CIA2 = $DD00      ;6526 CHIP BASE ADDR
  OUTPUT = $FF
  INPUT = $00
  OUTSHK = $04
  INTMSK = $90
  TOGHI = $04
  TOGLO = $FC
  NMIVEC = $0318
  ZPTEMP = $FB
  * = $C650
  START *=*+2       ;START ADDRESS
  END *=*+2         ;END ADDRESS
  MODE *=*+1        ;INPUT/OUTPUT FLAG
  VECTOR *=*+2      ;STORAGE FOR NMI VECTOR

  LDA MODE          ;INPUT OR OUTPUT
  BNE OUTDAT        ;BRANCH IF OUTPUT
  LDA #INPUT
  STA CIA2+3        ;SET DDR FOR INPUT
  LDA #INTMSK
  STA CIA2+13       ;FLAG INTERRUPTS DISABLED
  LDA CIA2+2
  ORA #OUTSHK
  STA CIA2+2        ;SET PA2 FOR OUTPUT
  LDA CIA2
  ORA #TOGHI
  STA CIA2          ;SET HANDSHAKE LINE PA2 HIGH
  LDA START
  STA ZPTEMP
  LDA START+1       ;MOVE POINTERS TO ZERO PAGE 0
  STA ZPTEMP+1
 ;
 ;INITIALISE INPUT WEDGE
 ;
  LDA NMIVEC
  STA VECTOR        ;SAVE OLD NMI VECTOR
  LDA NMIVEC+1
  STA VECTOR+1
  SEI               ;
  LDA #<NXTIN       ;
  STA NMIVEC        ;INSERT DATA-INPUT WEDGE
  LDA #>NXTIN       ;
  STA NMIVEC+1      ;
  CLI               ;
  RTS
 ;
 ;INITIALISE OUTPUT WEDGE
 ;
OUTDAT
  LDA #OUTPUT
  STA CIA2+3        ;SET DDR FOR OUTPUT
  LDA #INTMSK
  STA CIA2+13       ;FLAG INTS DISABLED
 ;
 ;
  LDA NMIVEC
  STA VECTOR
  LDA NMIVEC+1      ;SAVE OLD NMI VECTOR
  STA VECTOR+1
  SEI               ;
  LDA #<NXTOUT      ;
  STA NMIVEC        ;INSERT DATA-OUTPUT WEDGE
  LDA #>NXTOUT      ;
  STA NMIVEC+1      ;
  CLI               ;
  RTS
 ;
 ;
 ;INPUT DATA SERVICE ROUTINE
 ;
 ;
NXTIN
  LDA #INTMSK       ;CHECK ICR
  BIT CIA2+13       ;INT CAUSED BY FLAG?
  BEQ NOTCOM        ;NO.. NORMAL NMI
 ;
 ;OK BYTE ON PORT
 ;
  LDA CIA2+1        ;READ BYTE
  STA (ZPTEMP),Y    ;STORE IN MEMORY
  INC ZPTEMP
  BNE TEST1         ;INCREMENT POINTER
  INC ZPTEMP+1
 ;
TEST1
  LDA END
  CMP ZPTEMP
  LDA END+1         ;CHECK TO SEE IF ENDED
  SBC ZPTEMP+1
  BCC DONE          ;BRANCH IF FINISHED
 ;
 ;TELL DEVICE READY FOR NEXT BYTE
 ;
  LDA CIA2
  AND #TOGLO
  STA CIA2          ;TOGGLE PA2 LOW THEN HIGH
  ORA #TOGHI
  STA CIA2
 ;
 ;NOW DO NORMAL NMI ROUTINE
 ;
NOTCOM
  JMP (VECTOR)


 ;
 ;
 ;OUTPUT DATA SERVICE ROUTINE
 ;
 ;
NXTOUT
  LDA #INTMSK       ;CHECK ICR
  BIT CIA2+13       ;INTERRUPT CAUSED BY FLAG?
  BEQ NOTCOM        ;NO.. DO NORMAL NMI
 ;
 ;OK SEND BYTE
 ;
  LDA (ZPTEMP),Y    ;GET BYTE FROM MEMORY
  STA CIA2+1        ;OUTPUT IT. PC WILL GO
                    ;LOW FOR 1 CYCLE
  INC ZPTEMP
  BNE TEST2         ;INCREMENT POINTER
  INC ZPTEMP+1

TEST2
  LDA END
  CMP ZPTEMP
  LDA END+1         ;CHECK TO SEE IF ENDED
  SBC ZPTEMP+1
  BCC DONE          ;BRANCH IF DONE
 ;
 ;CONTINUE NORMAL NMI ROUTINE
 ;
  JMP (VECTOR)
 ;
 ;
 ;FINISHED REMOVE WEDGE
 ;
 ;
DONE
  SEI
  LDA VECTOR
  STA NMIVEC        ;RESET NMI VECTOR TO
  LDA VECTOR+1      ;ORIGINAL VALUE
  STA NMIVEC+1
  CLI
  JMP (NMIVEC)
```

Parawedge stammt aus ‚Mastering The Commodore 64' von Jones und Carpenter und wird mit freundlicher Genehmigung der Autoren und der Ellis Horwood Ltd. abgedruckt.

Fast MSX-Standard

Die amerikanischen Spectravideo-Computer 318 und 328 erfüllen einige Anforderungen des MSX-Standards. Spectravideo ist auf dem europäischen Markt relativ unbekannt, doch sind die Rechner wegen ihrer günstigen Preislage und ihrer Ausbaufähigkeit durchaus interessant für private Anwender.

Der Spectravideo 318 ist ein preiswerter Heimcomputer, der in mancher Hinsicht einem Vergleich mit dem Commodore 64 standhält. Er verfügt über einen dreistimmigen Synthesizer-Klangbaustein, hochauflösende Grafik einschließlich Sprites, einen eingebauten Joystick und einen Modulschacht.

Die Tastatur erfüllt fast die MSX-Spezifikationen. Sie entspricht vom Aufbau her der des Sinclair Spectrum und hat auch ähnliche Gummimembrantasten, ist aber großzügiger und solider angelegt, so daß sich besser damit arbeiten läßt.

Den Cursor-Tastenblock ersetzt ein integrierter Joystick. Sein Griff ist nur locker in eine Kippscheibe eingesteckt; zieht man ihn heraus, kann der Cursor durch direkten Druck auf Fingermulden in der Scheibe nach oben, unten, rechts oder links bewegt werden. Ebensogut können Sie natürlich den Joystick verwenden, etwa um den Cursor bei Programmkorrekturen schnell und sicher an die Fehlerstelle zu führen.

Das Spectravideo-BASIC ist eine Microsoft-Version, eng verwandt mit dem MSX-BASIC. Die BASIC-Programmierung wird durch den bildschirmorientierten Editor (mit automatischer Zeilennumerierung usw.) sehr erleichtert. Zwar fehlen „strukturierte" Befehle wie WHILE. . .WEND und REPEAT. . .UNTIL, die bei anderen BASIC-Fassungen zu finden sind, aber es gibt das vollständige IF. . . THEN. . .ELSE, das für eine übersichtliche und effiziente Programmierung unerläßlich ist. Das BASIC schöpft auch die Grafikmöglichkeiten des Rechners voll aus.

Die grafische Darstellung ist mit 256 × 192 Punkten und 16 Farben ganz brauchbar, obwohl die Farbbelegung eines einzelnen Pixels nicht möglich ist. Verglichen mit einigen ande-

Die Tastatur des 318 erfüllt nahezu die Anforderungen des MSX-Standards. Es handelt sich wie beim Spectrum um eine Gummimembranausführung, sie ist aber solider und weniger eng angelegt. Zur Ausstattung gehören neben den üblichen Control-, Escape-, Tab- und Backspace-Tasten fünf doppelt belegbare Funktionstasten, eine STOP- und eine SELECT-Taste sowie die Editiertasten INSert, DELete und COPY. Grafikzeichen werden durch Betätigung einer der Buchstabentasten in Verbindung mit der LEFT GRPH- oder der RIGHT GRPH-Taste aufgerufen.

ren Rechnern mag das etwas dürftig erscheinen, aber für die praktisch erreichbare Grafikqualität ist der Bedienkomfort eigentlich wichtiger als die Rasterspezifikation – und der 318 verfügt über alle Grafikerweiterungen des Microsoft-BASIC, etwa über Funktionsbefehle zum Zeichnen von Punkten, Linien, Kästchen, Kreisbögen und Ellipsen. Zum Füllen von Flächen dient das PAINT-Kommando, und mit speziellen VPOKE- und VPEEK-Anweisungen läßt sich der Bildspeicher direkt ansprechen. Außerdem gibt es eine Art „Grafiksprache", die in Verbindung mit dem DRAW-Befehl die Darstellung komplexer Formen und Zeichnungen unterstützt.

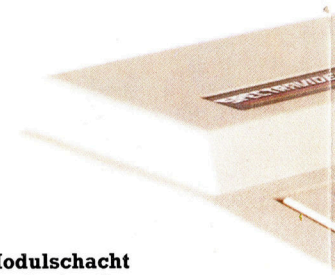
Beliebige rechteckige Bildschirmausschnitte können mit GET in ein BASIC-Array übertragen und mit PUT wieder auf den Schirm gebracht werden. Diese Befehle sind sehr nützlich für den Aufbau regelmäßiger Muster und für Spiegelungs- oder sogar einfache Bewegungseffekte. Zudem ermöglicht der Video-Chip 9929 die Definition von Sprites.

Die Anweisung ON SPRITE GOSUB erlaubt die Aufstellung einer „Ereignisfalle" (Event Trap). Das Programm läuft dann normal, aber sobald zwei Sprites kollidieren, verzweigt das System zu einer Interrupt-Routine. Auf diese Weise kann beispielsweise überwacht werden, ob ein Geschoß ein Raumschiff getroffen hat, ohne daß der Programmierer ständig alle möglichen Ereignisse im Auge behalten müßte. Das beschleunigt die Entwicklung und den Ablauf der Programme; ähnliche Ereignisfallen können auch für die Funktionstaste oder den Joystick installiert werden.

Für eine vielseitige Klangerzeugung verfügt der Soundchip des Spectravideo über drei Stimmen und eine Anzahl von Spezialeffekten. Damit lassen sich eindrucksvolle Ergebnisse erzielen, obwohl BASIC für aufwendigere Klangstrukturen nicht besonders geeignet ist. Die Wiedergabe erfolgt über ein Fernsehgerät, und dort ist auch die Lautstärke zu regeln.

Großer Bruder
Eine attraktive Alternative zum 318 ist der Spectravideo 328 – eine aufwendigere Version mit Schreibmaschinentastatur, 80 KByte RAM und eingebautem Textverarbeitungsprogramm. Er ist für den Ausbau zum Bürosystem besser geeignet.

Integrierter Joystick
Der eingebaute Joystick ersetzt die üblichen Cursorsteuerungstasten. Bei abgezogenem Handgriff läßt sich der Cursor auch durch Fingerdruck auf die Trägerplatte nach oben, unten, links oder rechts führen. Der Einbau des Joystick ist mehr als eine kosmetische Verbesserung, denn mit ihm lassen sich auch Diagonalbewegungen ausführen.

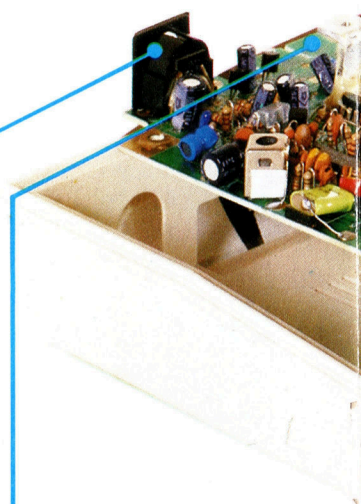


Modulschacht
Durch eine Aussparung werden Steckmodule in einen soliden Sockel gesetzt.

ROM
Das Spectravideo-BASIC ist in zwei ROMs zu je 16 KByte untergebracht.

Erweiterungsstecker
Hier läßt sich unter Zwischenschaltung einer Erweiterungsbox Zubehör anschließen.

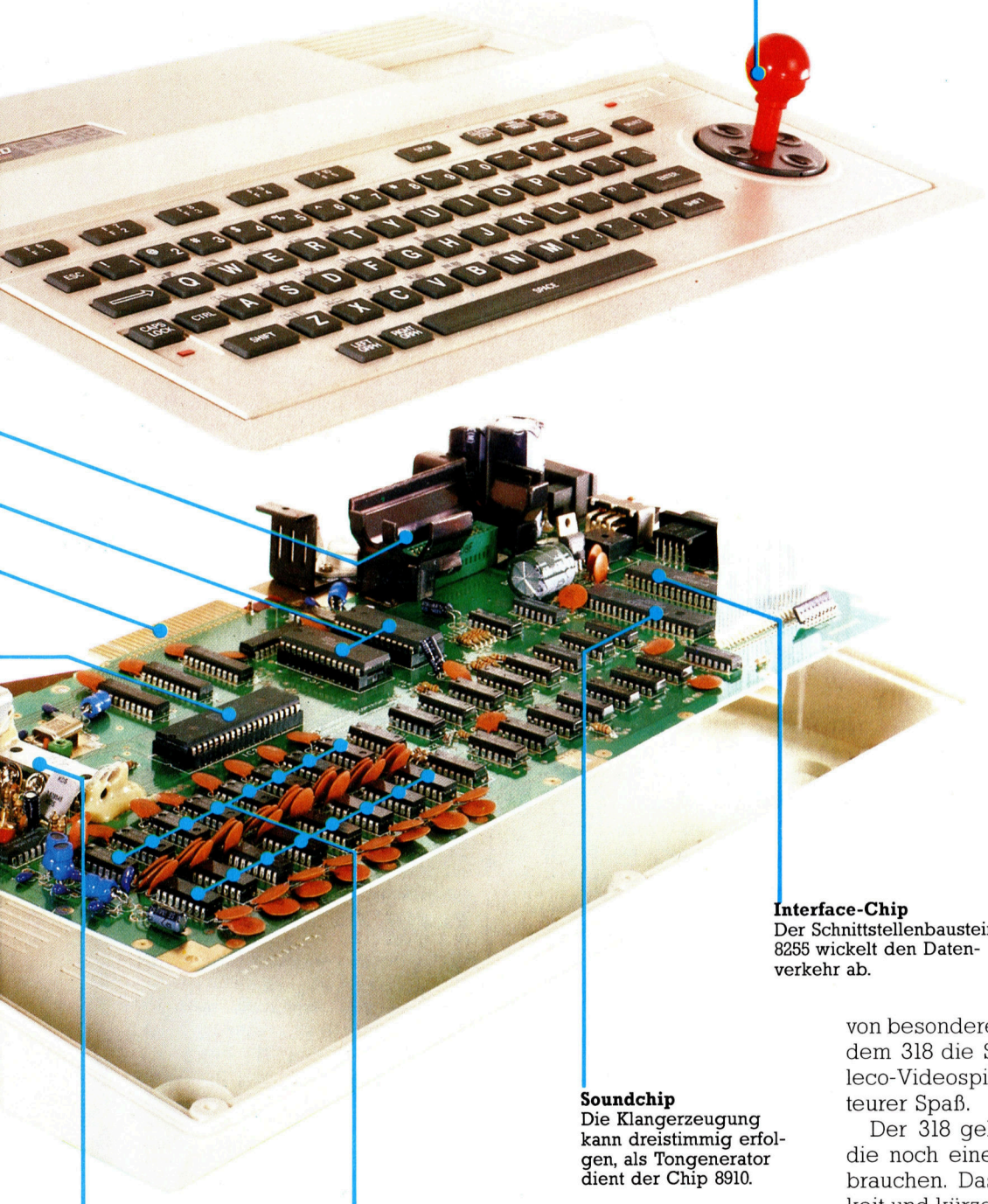
Microprozessor
Als CPU wurde der millionenfach verbreitete Z80 von Zilog verwendet.



Monitorausgang
Ein HF-Modulator ist beim 318 nicht eingebaut; die Verbindung mit dem Fernsehgerät muß daher extern erfolgen.

Der 318 weist Schnittstellen für zwei Joysticks, Cassettenrecorder und Steckmodule auf und hat eine Erweiterungs-Steckleiste. Es wird eine ganze Menge an Zubehör angeboten, allerdings nicht eben das billigste. Für dessen Anschluß ist der Erwerb eines „Mini-Expanders" erforderlich, der für genau ein Zubehörteil reicht – das ist zunächst meist eine 16- oder 64-KByte-Speichererweiterung. Für den weiteren Ausbau benötigt man den „Super-Expander" mit einem Steckleistensystem ähnlich wie beim Apple II; damit stehen zusätzlich sieben Anschlußmöglichkeiten für Speicherkarten, Druckerschnittstellen, Diskettenlaufwerke oder Modems zur Verfügung. Für Spiele-Fans dürfte der Coleco-Spieladapter

Cassettenrecorder-Anschluß
Am 318 ist nur der spezielle Spectravideo-Recorder verwendbar. Der Anschluß wird über einen Platinenstecker hergestellt.



Integrierter Joystick
Statt der üblichen Cursortasten wird beim 318 eine Kippscheibe verwendet, die einen abnehmbaren Griff trägt.

Interface-Chip
Der Schnittstellenbaustein 8255 wickelt den Datenverkehr ab.

Soundchip
Die Klangerzeugung kann dreistimmig erfolgen, als Tongenerator dient der Chip 8910.

RAM
Die Arbeitsspeicher-Kapazität von 32 KByte wurde in einer Doppelreihe von 16 RAM-Chips untergebracht.

Videobaustein
Der Videochip 9929 wird durch einen nachträglich hinzugefügten Kühlkörper verdeckt.

Spectravideo 318

ABMESSUNGEN:
410 × 220 × 80 mm

ZENTRALEINHEIT:
Z80

SPEICHER:
32 KByte RAM, davon etwa 12 K für BASIC-Programme verfügbar, 32 KByte ROM

BILDSCHIRMFORMAT:
24 Textzeilen zu 40 Zeichen, 80-Zeichen-Karte als Extra; Grafikauflösung maximal 256 × 192 Punkte bei 16 Farben, außerdem Sprite-Grafik

SCHNITTSTELLEN:
Erweiterungssteckleiste, Modulschacht, Cassettenrecorder-Anschluß, zwei Joystick-Eingänge, Monitorausgang

VERFÜGBARE SPRACHEN:
BASIC

TASTATUR:
Gummimembranausführung mit Funktions- und Editiertasten; eingebauter Joystick statt Cursortastenfeld

DOKUMENTATION:
Dürftige und fehlerhafte Anleitung.

STÄRKEN:
Viele der MSX-Spezifikationen werden erfüllt. Der Rechner verfügt über ein ausgezeichnetes BASIC, Sprite-Grafik und einen integrierten Joystick.

SCHWÄCHEN:
Mangelnde Software, keine serienmäßige Druckerschnittstelle, spezieller Cassettenrecorder erforderlich.

von besonderem Interesse sein. Damit läuft auf dem 318 die Steckmodul-Software für die Coleco-Videospiele, allerdings ist das ein relativ teurer Spaß.

Der 318 gehört zu den wenigen Rechnern, die noch einen speziellen Cassettenrecorder brauchen. Das bringt zwar mehr Zuverlässigkeit und kürzere Zugriffszeit, treibt aber insgesamt den Preis hoch.

Interessant ist der 318 vor allem wegen seiner Ausbaufähigkeit. Ist der Anwender bereit, in den Super-Expander, das 64-KByte-RAM, die 80-Zeichen-Karte und ein Diskettenlaufwerk zu investieren, so läßt sich das System durchaus mit einem kleinen CP/M-Bürocomputer vergleichen.

Aus Arm mach RAM

Nach der Kalibrierung kann der Digitalisier-Arm in Betrieb genommen werden. Wir führen ein Programm vor, das abgetastete Umrisse auf dem Bildschirm des Acorn B darstellt.

Der erste Schritt zur Programmentwicklung ist das Einmessen der Hardware. Das Kalibrierprogramm nutzt dazu vier Potentiometer-Meßwerte bestimmter Arm-Positionen: Nullstellung (armzero) des Oberarms (arm1) und des Unterarms (arm2) sowie die 90-Grad-Stellung (armninety) von Oberarm und Unterarm. Diese Werte werden in die Zeilen 1200, 1210, 1240 und 1250 des Programms als Teil der „define_parameters"-Routine eingegeben. Zwei weitere Parameter können von Tracer zu Tracer verschieden sein – die Längen der Armteile (length). Sie werden in den Zeilen 1220 und 1260 festgehalten. Beachten Sie, daß die Eingabe in Millimetern erfolgen soll. Beim Unterarm muß bis zum Fadenkreuz auf dem Visier gemessen werden.

Wenn die vier Kalibrierwerte bekannt sind, lassen sich damit die vom eingebauten Analog/Digitalwandler des Acorn B digitalisierten Meßwerte der Potentiometer in Winkel umrechnen:

```
angle = (ADVAL(n) — armzero)*90/
        (armninety — armzero)
```

Da der zweite Teil des Ausdrucks eine Konstante (factor) ist, wird er nur einmal berechnet und später bei allen Berechnungen erneut verwendet. Mit der Umrechnung

```
factor = 90/(armninety—armzero)
```

läßt sich die Formel dann so schreiben:

```
angle=(ADVAL(n)—armzero)*factor
```

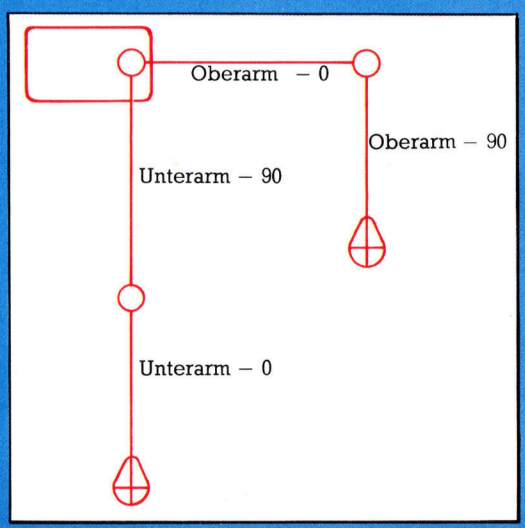
Kalibrierung

Für die Kalibrierung benötigen wir vier Eichwerte: Die 0- und 90-Grad-Stellungen des Oberarms und die 0- und 90-Grad-Positionen des Unterarms. Wenn das Kalibrierprogramm läuft, richten Sie dazu den Oberarm parallel zur Oberkante der Grundplatte, den Unterarm parallel mit der seitlichen Kante aus.

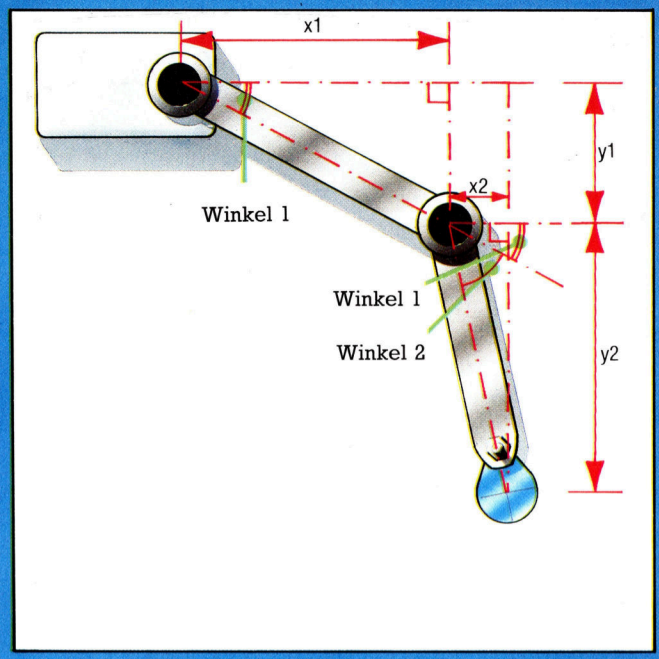
Schreiben Sie die Werte auf. Als nächstes beide Armteile parallel zur seitlichen Kante der Grundplatte ausrichten und erneut die Werte notieren. Durch einen Tastendruck gehen Sie nun in den angle-Modus.

Die vier aufgeschriebenen Werte müssen jetzt – auf die entsprechenden Bildschirmmeldungen hin – eingetippt werden. Prüfen Sie, ob die angezeigten Winkel für beide Arme stimmen! Falls nicht, müssen Sie noch einmal von vorne beginnen. Verwahren Sie den Zettel mit den notierten Werten – er wird für das Steuerprogramm noch einmal gebraucht.

Eichpositionen




```
1000 REM **** BBC TRACER CALIBRATION ****
1010 PROCcalibrate
1020 PROCdisplay_angle
1030 END
1040 DEF PROCcalibrate
1050 CLS
1060 PRINT TAB(5,10);"ANGLE 1";TAB(20);"ANGLE 2"
1070 REPEAT
1080 PRINT TAB(5,12);SPC(40)
1090 PRINT TAB(5,12);ADVAL(1);TAB(20)ADVAL(2)
1100 PROCdelay(600)
1110 A$=INKEY$(1)
1120 UNTIL A$<>""
1130 ENDPROC
1150 DEF PROCdisplay_angle
1160 PROCdefine_angles
1170 CLS
1180 PRINT TAB(5,10);"ANGLE 1";TAB(20);"ANGLE 2"
1190 REPEAT
1200 PROCcalc_angles
1210 PRINT TAB(5,12);SPC(40)
1220 PRINT TAB(5,12);angle1;TAB(20)angle2
1230 PROCdelay(1000)
1240 A$=INKEY$(1)
1250 UNTIL A$<>""
1260 ENDPROC
1280 DEF PROCdefine_angles
1290 CLS
1300 PRINT TAB(5,10);:INPUT"1st arm zero position"
;one_zero
1310 PRINT TAB(5);:INPUT"1st arm ninety position"
;one_ninety
1320 PRINT TAB(5);:INPUT"2nd arm zero position"
;two_zero
1330 PRINT TAB(5);:INPUT"2nd arm ninety position"
;two_ninety
1340 one_factor=90/(one_ninety-one_zero)
1350 two_factor=90/(two_ninety-two_zero)
1360 ENDPROC
1380 DEF PROCcalc_angles
1390 angle1=INT((ADVAL(1)-one_zero)*one_factor)
1400 angle2=INT((ADVAL(2)-two_zero)*two_factor)
1420 DEF PROCdelay(delay)
1430 LOCAL I
1440 FOR I=1 TO delay:NEXT I
1450 ENDPROC
1460 PROCdelay(1000)
```



Geometrie des Arms

Die horizontale und vertikale Verschiebung des Arms kann aus den Meßwerten der beiden Potentiometer ermittelt werden. Die dafür notwendigen geometrischen Berechnungen basieren auf den beiden eingezeichneten rechtwinkligen Dreiecken. Die Werte x1 und y1 entsprechen der Länge des Oberarms multipliziert mit dem Cosinus bzw. Sinus von Winkel 1. Der Meßwert des zweiten Potentiometers wird zusätzlich durch die Position des Oberarms beeinflußt, daher sind die Werte x2 und y2 etwas anders zu berechnen. Der tatsächliche Winkel ist nicht Winkel 2, sondern Winkel 1 + Winkel 2. Cosinus- und Sinuswert des berechneten Winkels ergeben multipliziert mit der Länge des Unterarms die horizontale und vertikale Verschiebung. Addition von x1 und x2 bzw. y1 und y2 gibt die Visierposition an.

Das Unterprogramm calc_xy ermittelt aus den aktuellen Winkelmeßwerten der beiden Potentiometer die Verschiebung des Fadenkreuzes in bezug auf die x- und y-Achse. Die dabei verwendeten geometrischen Verfahren sind nicht übermäßig kompliziert. x1 und y1 stehen mit der Länge des Oberarms in einem durch Sinus- und Cosinusfunktionen beschreibbaren Verhältnis:

```
x1=COS(angle1)*length_arm1
y1=SIN(angle1)*length_arm1
```

Die Berechnung von x2 und y2 geschieht etwas anders, weil hier auch die Stellung des Oberarms bedeutsam ist:

```
x2 = COS(angle1 + angle2)*length_arm2
y2 = SIN(angle1 + angle2)*length_arm2
```

Die gesamte x- und y-Verschiebung ergibt sich durch Addition der gültigen Werte von x1 und x2 bzw. y1 und y2.

Die im Unterprogramm calc_xy verwendeten Formeln weichen von den obigen ein wenig ab. Die Variablen der Armlängen sind darin so umgerechnet, daß sich die Werte direkt im Grafik-Koordinatensystem des Acorn B verwenden lassen.

Die beste Methode zum Speichern des Bildschirminhalts ist, mit einem Maschinenprogramm den in MODE 1 verwendeten RAM-Bereich zu speichern. Wenn der Bildschirm seit dem letzten CLS nicht mehr gescrollt worden ist, liegt dieser Bereich zwischen &3000 und &7FFF. Problematisch ist allerdings, daß der Filename nicht direkt als Stringvariable an den *SAVE-Befehl übergeben werden kann. Der nachfolgende Befehl

```
*SAVE file$ 3000 8000
```

speichert die Datei mit dem Namen file$, nicht aber den String in der Variablen file$. Der Bildschirminhalt kann daher nicht ohne weiteres unter verschiedenen Filenamen auf Cassette oder Diskette abgespeichert werden. Mit dem Betriebssystemaufruf OSCLI kann man das Problem glücklicherweise umgehen: Dieser Befehl führt einen im Speicher befindlichen Block mit ASCII-Codes genau so aus, als ob die entsprechenden Zeichen direkt über die Tastatur eingegeben worden wären. Dazu müssen wir die Startadresse des ASCII-Blocks im Lo-Byte/Hi-Byte-Format in die Register X und Y schreiben. Zur Ausführung von *SAVE und *LOAD wird der Befehl (zusammen mit dem gewählten Filenamen) zuerst in einen String assembliert. Danach werden die ASCII-Werte der Zeichen dieses Strings in einen freien Speicherbereich geschrieben. Wenn nun das X- und das Y-Register auf die Startadresse dieses Bereichs zeigen, wird der gespeicherte Befehl durch Aufruf von OSCLI an Adresse &FFF7 ausgeführt. Unser Programm enthält für diesen Zweck in Zeile 2170 die Routine „oscli_command".

Neue Farben einstellen

Der Bildschirminhalt kann durch Drücken von C gelöscht werden. Mit M springt das Programm ins Hauptmenü zurück. Die zweite Menüoption können Sie jetzt noch nicht anwählen – dazu wird der nächste Programmteil gebraucht, den wir im folgenden Abschnitt bringen. Neue Farben für den Vordergrund lassen

sich jederzeit mit den Tasten 1, 2 oder 3 einstellen. Diese Farben sind die im Mode 1 voreingestellten Farben und können durch den Befehl VDU 19 geändert werden.

Mit der hier vorgestellten ersten Programmhälfte können Sie den Arm frei führen (Freihand-Modus). Mit dem zweiten Programmteil wird auch die Definition von Punkten, Strichen und Kreisen möglich. Im Freihand-Modus sind sechs Funktionen verfügbar: Mit dem am Digitalisiertablett montierten Taster können Sie den „Stift" aufsetzen und Linien ziehen (draw) oder ihn hochheben und ohne Linie auf dem Bildschirm bewegen (move). Der Taster ist mit den Joystick-Feuerknopf-Anschlüssen des Analogport verbunden und wird mit Hilfe der beiden niederwertigen Bits im ADVAL(0)-Register abgefragt. Wird einer der beiden Feuerknöpfe gedrückt, geht das entsprechende Bit auf High (1). Bei (ADVAL(0)AND3)<>0 wurde ein Feuerknopf gedrückt, und die entsprechende Reaktion erfolgt prompt. In unserem Programm wird dadurch die Variable „togflag" von 0 auf 1 (oder umgekehrt) gesetzt. Das Unterprogramm „draw" fragt „togflag" ab und entscheidet entsprechend dem gespeicherten Wert, ob der nächste Punkt mit DRAW oder mit MOVE angesteuert wird.

Kreuz im Visier

Auf dem Bildschirm wird die Visierposition durch ein kleines Kreuz dargestellt. Durch den „Exclusive OR"-Zeichenmodus kann es ohne Beeinträchtigung der Darstellung bewegt und auch gelöscht werden. Jede Linie kann in diesem durch GCOL3 angewählten Modus auch wieder beseitigt werden.

```
   6 REM ** BBC DIGITAL TRACER **
 1080 PROCdefine_parameters
 1090 REPEAT
 1100 MODE 1
 1110 REM CURSOR OFF
 1120 VDU 23,1,0;0;0;0;
 1130 PROCmenu
 1140 CLS
 1150 IF ans$="1" THEN PROCfreehand ELSE PROCelastic
 1160 UNTIL endflag=1
 1170 END
```

Im Programm sind für die Länge der Armteile und Potentiometer-Meßwerte bereits Daten vorgegeben.

```
 1190 DEF PROCdefine_parameters
 1200 one_zero=14820
 1210 one_ninety=45020
 1220 one_length=250
 1230 one_factor=90/(one_ninety-one_zero)
 1240 two_zero=450
 1250 two_ninety=25300
 1260 two_length=222+28
 1270 two_factor=90/(two_ninety-two_zero)
 1280 REM ** MM TO GRAPHICS COORDS CONVERSION FACT
ORS **
 1290 scale=1023/460
 1300 one_scale=scale*one_length
 1310 two_scale=scale*two_length
 1320 REM SCAN 2 ADC CHANNELS ONLY
 1330 *FX 16,2
 1340 endflag=0:togflag=0:colour=1
 1350 DIM save% 30:DIM x(3),y(3)
 1360 ENDPROC
 1380 DEF PROCfreehand
 1390 PROCfree_inform
 1400 REPEAT
 1410 IF (ADVAL(0) DIV 256)<>0 THEN PROCdraw
 1420 IF (ADVAL(0) AND 3)<>0 THEN PROCtoggle_pen
 1430 ans$=INKEY$(1):IF ans$<>"" THEN PROCfreepress
 1440 UNTIL exitflag=1
 1450 ENDPROC
 1470 DEF PROCdraw
 1480 PROCcalc_xy
 1490 PROCcursor(oldx,oldy)
 1500 MOVE oldx,oldy
 1510 IF togflag=0 THEN DRAW x,y ELSE MOVE x,y
 1520 PROCcursor(x,y)
 1530 oldx=x:oldy=y
 1540 ENDPROC
 1560 DEF PROCcursor(cx,cy)
 1570 GCOL 3,3:REM EOR PLOT MODE
 1580 MOVE cx,cy-16
 1590 PLOT 1,0,32
 1600 PLOT 0,-16,-16
 1610 PLOT 1,32,0
 1620 GCOL 0,colour
 1630 ENDPROC
```

Dieser Programmteil rechnet die Potentiometer-Meßwerte in die horizontalen und vertikalen Komponenten x und y um.

```
 1650 DEF PROCcalc_xy
 1660 angle1=RAD((ADVAL(1)-one_zero)*one_factor)
 1670 angle2=RAD((ADVAL(2)-two_zero)*two_factor)
 1680 x=one_scale*COS(angle1)+two_scale*COS(angle1
+angle2)
 1690 y=1023-(one_scale*SIN(angle1)+two_scale*SIN(
angle1+angle2))
 1700 ENDPROC
 1720 DEF PROCmenu
 1730 exitflag=0:endflag=0
 1740 PRINT TAB(5,10);"Please choose"
 1750 PRINT TAB(5);"1...Freehand mode"
 1760 PRINT TAB(5);"2...Elastic mode"
 1770 PRINT:PRINT TAB(5);"Press 1 or 2"
 1780 REPEAT:ans$=GET$:UNTIL ans$="1" OR ans$="2"
 1790 ENDPROC
 1810 DEF PROCtoggle_pen
 1820 togflag=1-togflag
 1830 REPEAT UNTIL (ADVAL(0) AND 3)=0
 1840 ENDPROC
 1860 DEF PROCfreepress
 1870 IF ans$="C" THEN CLS:PROCfree_inform:ENDPROC
 1880 IF ans$="M" THEN exitflag=1
 1890 IF ans$="S" THEN PROCsave_screen:PROCfree_in
form:ENDPROC
 1900 IF ans$="L" THEN PROCload_screen:PROCfree_in
form:ENDPROC
 1910 PROCcolour_change
 1920 ENDPROC
 1940 DEF PROCfree_inform
 1950 PROCcalc_xy:oldx=x:oldy=y:PROCcursor(oldx,oldy)
 1960 PRINT TAB(1,1);SPC(79)
 1970 PRINT TAB(1,1);"S=Save  L=Load  M=Menu  C=Clear"
 1980 GCOL 0,1:MOVE 0,920:DRAW 1280,920
 1990 ENDPROC
 2010 DEF PROCcolour_change
 2020 ans=VAL(ans$)
 2030 IF ans<1 OR ans>3 THEN ENDPROC
 2040 colour=ans
 2050 ENDPROC
```

Dieser Programmteil liest den Bildschirminhalt in einen RAM-Bereich, der durch Aufruf von OSCLI geladen und gespeichert werden kann.

```
 2070 DEF PROCsave_screen
 2080 PROCcursor(x,y):REM CURSOR OFF
 2090 REPEAT
 2100 PRINT TAB(1,1);SPC(79)
 2110 INPUT TAB(1,1);"SAVE FILENAME";file$:file$=f
ile$+".S"
 2120 UNTIL LEN(file$)<8
 2130 file$="*SAVE "+file$+" 3000 8000"
 2140 PROCoscli_command(file$)
 2150 ENDPROC
 2170 DEF FROCoscli_command(a$)
 2180 FOR I=0 TO LEN(a$)-1
 2190 save%?I=ASC(MID$(a$,I+1,1))
 2200 NEXT I
 2210 save%?I=13:REM ADD CR
 2220 X%=save% MOD 256:Y%=save% DIV 256
 2230 CALL &FFF7:REM CALL OSCLI
 2240 ENDPROC
 2260 DEF PROCload_screen
 2270 REPEAT
 2280 PRINT TAB(1,1);SPC(79)
 2290 INPUT TAB(1,1);"LOAD FILENAME";file$:file$=f
ile$+".S"
 2300 UNTIL LEN(file$)<8
 2310 file$="*LOAD "+file$
 2320 PROCoscli_command(file$)
 2330 ENDPROC
```

Farbenfreude

Wir untersuchen, wie über die Register des VIC-II des Commodore 64 unterschiedliche Darstellungsmodi erzeugt werden. Wir generieren Farbdarstellungen und zeigen, wie der VIC-II den Speicher anspricht.

Auf dem Commodore 64 gibt es acht grundlegende Grafikmodi. In niedriger Auflösung befindet sich der Zeichensatz im RAM oder ROM und kann auf drei verschiedene Weisen angezeigt werden: Standardmodus, Vielfarbmodus und erweiterter Farbmodus. Weiterhin gibt es zwei Arten von hoher Auflösung: Standardmodus und Vielfarbmodus. Außer den Grundarten sind weitere Variationen möglich: Der Bildschirm kann auf 38 Spalten (statt der normalen 40) gesetzt werden und/oder auf 24 Zeilen (Standard sind 25). Die letzten beiden Darstellungsarten werden normalerweise mit dem „sanften" horizontalen und vertikalen Scrollen des Bildschirms eingesetzt, das auf dem Commodore 64 am besten vom Maschinencode aus gesteuert wird.

Wenn ein langes Programm mit hochauflösender Grafik arbeitet, wird leicht der Speicherplatz knapp. Der Bildschirmspeicher für niedrige und hohe Auflösung läßt sich jedoch an fast jede Position des Arbeitsspeichers verlegen. Wir sehen uns daher zunächst an, wie eine Verlegung durchgeführt wird. Wenn Ihr Programm ausschließlich Maschinencode enthält, können Sie den Bildschirmspeicher für hohe Auflösung sogar „hinter" das ROM des BASIC-Interpreters legen und so die acht KByte des Interpreter-ROMs besser einsetzen. Interessanterweise benötigt die hochauflösende Grafik gerade acht KByte des vorhandenen Speicherplatzes.

Auf Empfang

Der 6555/67 Video-Interface-Chip (VIC-II) hat die Aufgabe, Bildschirmdaten zu erzeugen, die dann an den Fernseher oder Monitor weitergeleitet werden. Der VIC-II muß dazu Daten aus dem RAM oder ROM lesen. Um das recht komplizierte Verhalten des C64 besser verstehen zu können, untersuchen wir zunächst, wie der VIC-II Daten erhält.

Im Vielfarbmodus kann ein Zeichenfeld vier statt zwei Farben haben. Damit das möglich ist, besteht die horizontale Auflösung hier nicht mehr aus einzelnen Pixeln, sondern aus Pixelpaaren. Der Vielfarbmodus läßt sich im hohen oder niedrigen Grafikmodus einsetzen, wenn auch die Pixelfarben bei der hohen Auflösung auf andere Weise bestimmt werden. Die folgenden BASIC-POKEs schalten den Vielfarbmodus an und ab:

```
POKE 53270,PEEK(53270)OR16
POKE 53270,PEEK(53270)AND239
```

Wenn bei niedriger Auflösung und Vielfarbmodus das Bit 4 des entsprechenden Farbnybbles (ein Nybble ist ein Halbbyte oder vier Bits) auf Eins steht, wird das Zeichen im Vielfarbmodus interpretiert, wobei die niederwertigen drei Bits die Farbe festlegen. Zeichen mit Farbnybbles im Bereich von 0 bis 7 werden daher normal interpretiert, Zeichen mit Farbcode zwischen 8 und 15 hingegen im Vielfarbmodus. Tabelle 1 zeigt, wie die Farben der Pixelpaare festgelegt werden.

Erweiterter Farbmodus

Durch Austauschen der Adressen 53282 und 53282 können Sie im Handumdrehen die Farbe jedes Pixelpaares wechseln. Der Vielfarbmodus eignet sich besonders für anwenderdefinierte Zeichen, da hier beim Bitmuster die Pixelpaare berücksichtigt wurden.

Der „erweiterte Farbmodus" bietet zusätzliche Darstellungsmöglichkeiten. Diese Farbart steuert die Hintergrundfarbe der ersten 64 Zeichen der Zeichenmatrix. Der erweiterte Farbmodus läßt sich jedoch nicht zusammen mit dem Vielfarbmodus einsetzen. Je eine BASIC-Zeile schaltet den erweiterten Farbmodus an bzw. ab:

```
POKE 53265,PEEK(53265)OR64
POKE 53265,PEEK(53265)AND191
```

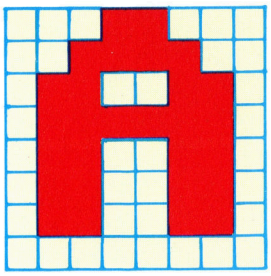
Auf dem Bildschirm kann nun das gleiche Zeichen in vier verschiedenen Hintergrundfarben erscheinen (eine davon ist die des Bildschirms). Die verbleibenden Zeichen lassen

Tabelle 1

Bitmuster des Pixelpaares		Farbe	Festgelegt durch
0	0	Bildschirmhintergrund	53281 ($D021) Bits 0 bis 4
0	1	Vielfarbmodus #1	53282 ($D022) Bits 0 bis 4
1	0	Vielfarbmodus #2	53283 ($D023) Bits 0 bis 4
1	1	Vordergrundfarbe	Bits 0 bis 3 des Farbnybbles

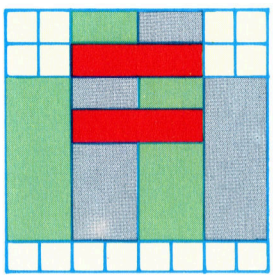
Tabelle 2

Bildschirmcode	Bit 7	Bit 6	Register für Hintergrundfarbe
64–127	0	1	53282 ($D022) Erweiterte Farbe 1
128–191	1	0	53283 ($D023) Erweiterte Farbe 2
192–255	1	1	53284 ($D024) Erweiterte Farbe 3



„A" im normalen Anzeigemodus

0	0	0	1	1	0	0	0
0	0	1	1	1	1	0	0
0	1	1	0	0	1	1	0
0	1	1	1	1	1	1	0
0	1	1	0	0	1	1	0
0	1	1	0	0	1	1	0
0	1	1	0	0	1	1	0
0	0	0	0	0	0	0	0



„A" im Vielfarbmodus

00	01	10	00
00	11	11	00
01	10	01	10
01	11	11	10
01	10	01	10
01	10	01	10
01	10	01	10
00	00	00	00

Farbschlüssel

01 Vielfarbmodus 1

10 Vielfarbmodus 2

11 Vordergrundfarbe

00 Hintergrundfarbe

Im normalen Anzeigemodus des C 64 wird jedes Bit eines Zeichens, das auf Eins steht, in Vordergrundfarbe dargestellt und jedes Bit, das auf Null steht, in Hintergrundfarbe. Beim Vielfarbmodus werden die Bitmuster jedoch als Bitpaare interpretiert. Die vier möglichen Kombinationen eines Bitpaares stellen außer den Farben des Vorder- und Hintergrundes noch zwei zusätzliche Farben dar.

sich im erweiterten Farbmodus nicht darstellen, da Bit 6 und 7 des Bildschirmcodes für die indirekte Steuerung der Zeichenfarbe gebraucht wird. So ist 1 beispielsweise der Bildschirmcode für „A" und 65 der für „A" in Negativdarstellung. Wenn Sie jedoch 65 in den Bildschirmspeicher POKEn, sehen Sie kein inverses „A", sondern ein normales „A", dessen Hintergrundfarbe der Inhalt der Adresse 53282 ($D022) bestimmt. Auf die gleiche Weise erzeugt ein POKE 129 in den Bildschirmspeicher ein normales „A", jedoch mit der Hintergrundfarbe, die die Adresse 53283 ($D023) angibt. Tabelle 2 zeigt die Beziehung zwischen Bildschirmcodes und Farbregistern.

Der VIC-II Chip arbeitet mit einem einfachen Speicherschema. Er kann jeweils nur einen von vier Blöcken zu je 16 KByte – Bank genannt – „sehen". Stellen Sie sich diese Banks am besten als Fenster des VIC-II zum Arbeitsspeicher vor. Die Basisadresse des Fensters ist softwaregesteuert und kann einen von vier Werten annehmen.

```
1000 REM**BANK FUER DAS VIC FENSTER
     WAEHLEN**
1010 POKE 56578,PEEK(56578)OR3:REM
     DDR
     CIA#2 BITS 0,1 ZUR AUSGABE
1020 WD=3:REN WAEHLT NORMALES
     FENSTER
1030 POKE 56576,(PEEK(56576)AND252)OR
     WD:
     REM PORT A CIA#1 BITS 0,1
```

WD kann Werte von 0 bis 3 annehmen. Der aktuelle Wert von WD läßt sich jederzeit mit PEEK(56576)AND3 abfragen. Die Adresse der Untergrenze des Speicherfensters wird mit der Formel WB=16384*(3-WD) errechnet, wobei WD die entsprechenden Adressen liefert:

WD	Fenster Startadresse
0	49152 ($C000)
1	32768 ($8000)
2	16384 ($4000)
3	0 ($0000)

Im Inneren des VIC-II Fensters erwartet der 6510 den Bildschirmspeicher und ein Abbild des Zeichen-ROMs in niedriger Auflösung (oder in hoher, wenn dieser Grafikmodus angeschaltet ist). Die CPU findet dort auch ihre Sprite-Daten. Die Pointer auf die acht Sprites sind am Ende des Bildschirmspeichers untergebracht und müssen bei einer Verlegung ebenfalls verschoben werden.

Die vier höherwertigen Bits des VIC-II Steuerregisters bei der Adresse 53272 ($D018) geben an, welches Offset auf die Basis des aktuellen VIC-II Fensters den Anfang des Bildschirmspeichers angibt. Mit diesen vier Bits kann der Bildschirmspeicher in jeden der 16K-Fensterblöcke gelegt werden.

```
1040 REM**OFFSET DES BILDSCHIRMS VON
     DER BASIS DES FENSTERS
     WAEHLEN**
1050 SO=1:REM LEGT NORMALES OFFSET
     FEST
1060 POKE 53272,(PEEK(53272)AND15)
     OR16*SO
```

SO kann hier Werte von 0 bis 15 annehmen. Der aktuelle Wert von SO kann jederzeit mit PEEK(53272)AND15 festgestellt werden. Die entsprechende Basisadresse des aktuellen Bildschirmspeichers wird entweder mit SC=WB+1024*SO (Fensterbasis plus Offset) errechnet oder mit:

```
SC=16384*(3-(PEEK(56576)AND3))+64*(PEEK
53272)AND240)
```

Da das Farb-RAM sich bei der Verlegung des Bildschirmspeichers nicht mitbewegt, gibt es eine Schwierigkeit. Für einen kompletten zweiten Bildschirmspeicher brauchen Sie eine kurze Maschinencoderoutine, die den Farbspeicher in und aus einem Buffer bewegt.

Zeigereinstellung

Wenn Sie den neuen Bildschirm mit PRINT ansprechen wollen, müssen Sie weiterhin dem Betriebssystem mitteilen, wo sich der neue Bildschirmspeicher befindet. Dies kann mit einem POKE (oder STA) auf Adresse 648 ($0288) geschehen, da dort der Pointer auf das höherwertige Byte der Bildschirmspeicher-Basisadresse zeigt. Wenn SC wie beschrieben berechnet wurde, führt

```
POKE 648,INT(SC/256)
```

diese Aufgabe aus. Die Basisadresse der Zeichenmatrix oder des hochauflösenden Bildschirmes wird folgendermaßen festgelegt:

```
1070 REM** HOCHAUFLOESENDES/CM
     OFFSET FUER DIE FENSTERBASIS
     WAEHLEN**
1080 HO=4:REM NORMALES OFFSET
     FESTLEGEN
1090 POKE 53272,(PEEK(53272)AND240)OR
     2*HO
```

Prinzipiell kann HO jeden Wert von 0 bis 7 annehmen; in der Praxis sind die Möglichkeiten jedoch eingeschränkt. Wenn WD 1 oder 3 entspricht, darf HO nicht die Werte 2 oder 3 annehmen, da der VIC-II dort das normale ROM-Zeichenabbild erwartet. Zu hohe Werte von HO verlegen die Obergrenze des hochauflösenden Speichers außerhalb des Bereichs, den der VIC-II ansprechen kann. Die entsprechende Basisadresse der aktuellen Zeichenmatrix (bzw. des hochauflösenden Bildschirmspeichers) wird entweder mit CM=WB+2048*HO (Fensterbasis plus Offset) errechnet, oder mit

```
CM=16384*(3-(PEEK(56576)AND3))+1024*
PEEK(53272)AND14)
```

Durch Setzen dieser Register lassen sich die Bildschirmspeicher in Bereiche verlegen, die sich für Ihr Programm am besten eignen.

Go total

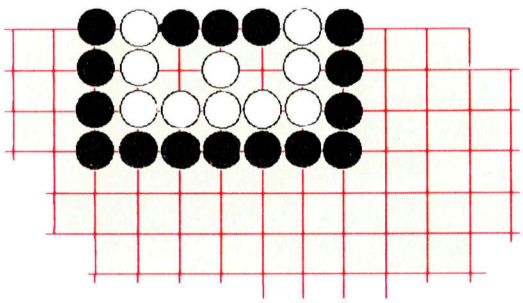
Wir setzen unser Go-Programmprojekt mit Listings der E/A-Routinen für den Commodore 64, Sinclair Spectrum und Schneider CPC 464/664 fort. Das Spielbrett wird generiert, und Unterroutinen ermöglichen die Ausgabe von Mitteilungen.

Soweit möglich entsprechen die hier gezeigten Listings für den Commodore 64, Sinclair Spectrum und Schneider CPC 464/664 dem bereits für den Acorn B vorgestellten Programm. Daher haben auch viele der im letzten Artikel gegebenen Erklärungen Gültigkeit.

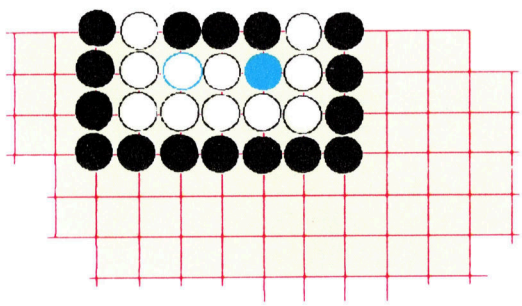
Um die Sache einfach zu halten, wurden in allen Programmversionen dieselben Zeilennummern verwendet. Nur bei größeren Unterschieden der maschinenspezifischen Verarbeitung und bei Abweichungen im BASIC wurden die Original-Algorithmen verändert. Der Hauptunterschied zwischen der Acorn-B-Version und den hier gezeigten Versionen, einmal abgesehen von der differierenden Bildschirmdarstellung der Systeme, besteht darin, daß die hier behandelten Rechner keine Prozeduren und Mehrfachfunktionen ermöglichen. Daher mußten die fehlenden Möglichkeiten durch Unterroutinen simuliert werden. Das sieht zwar etwas umständlich aus, funktioniert aber letztlich genausogut.

Seki

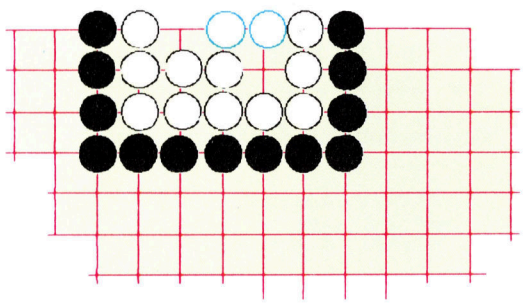
Wer schließt wen ein?
Die weiße Gruppe ist vollständig von größeren schwarzen Gruppen eingeschlossen und hat nur noch zwei Freifelder. Trotzdem schließt diese Gruppe eine kleinere schwarze Gruppe mit drei Steinen ein.



Fataler Fehler
Setzt Schwarz in der Absicht, die weiße Gruppe zu erobern, auf eines der zwei verbleibenden weißen Freifelder, kann Weiß auf das letzte Freifeld setzen und die schwarze Gruppe so vom Spielbrett entfernen.



Sicherheit
Jetzt kann sich die weiße Gruppe dadurch absichern, daß sie zwei Augen bildet. Schwarz hätte die beiden ursprünglich geteilten Freifelder unbesetzt lassen sollen – eine Situation, die bei Go als „Seki" bezeichnet wird.



Modul Eins:

Commodore 64

```
10 POKE 56578,PEEK(56578)OR3:POKE 56576,
(PEEK(56576)AND252)OR1
15 POKE648,132:POKE 52,128:POKE 56,128
20 DIM SK%(300)
30 POKE 53280,0:POKE 53281,0:PRINT"[CLEA
R][DOWN][DOWN][DOWN][DOWN][DOWN][DOWN][D
OWN][DOWN][DOWN][DOWN]                [WHITE]
PLEASE WAIT":GOSUB 170
40 GOSUB 1270
50 PRINT"[CLEAR]":GOSUB 1730
60 MOVE%=MOVE%+1:REM WHITE MOVES HERE
70 IF FIN% GOTO 100
80 MOVE%=MOVE%+1:REM BLACK MOVES HERE
90 IF NOT FIN% GOTO 60
100 IP%=22:IM%=9:IW%=1:GOSUB 1990
110 IF A$="Y" GOTO 40
115 IF A$<>"N" GOTO 100
120 MP%=24:MM%=5:O$="":GOSUB 2160
130 GOTO130
170 REM INITIALISE ROUTINE
190 BLACK%=1:WHITE%=2:COLOUR%=3
200 MARKER%=4:LIBERTY%=8
220 BOARD=49152
240 DIM CAPTURE%(2)
250 AXIS$="[c 3]ABCDEFGHIJKLMNO"
260 DIM DIR%(4)
270 FOR L=1 TO 4
280 READ DIR%(L)
290 NEXT
300 DATA 16,1,-16,-1
310 GOSUB 390
350 RETURN
390 REM READ-MESSAGES ROUTINE
420 DIM MESS$(9)
430 FOR M=0 TO 9
440 READ MESS$(M)
450 NEXT
460 DATA "O.K. THINKING..."
470 DATA "ILLEGAL ENTRY: "
480 DATA "STONE ALREADY ON POINT: "
490 DATA "ILLEGAL. KO ON POINT: "
500 DATA "ILLEGAL SUICIDE ON POINT: "
510 DATA "O.K. GAME OVER. PRESS RUN/STOP
"
520 DATA ""
530 DATA "  HOW MANY HANDICAP STONES MAY
 I          HAVE (2-9) ?"
```

```
540 DATA "TYPE YOUR MOVE (EG. H8), PASS
OR        QUIT :"
550 DATA "DO YOU WANT TO PLAY AGAIN (Y/N
)?"
560 RETURN
1270 REM INTRODUCTION ROUTINE
1280 GOSUB 1340
1290 GOSUB 1450
1340 REM GAME-INIT ROUTINE
1345 STACK%=1
1360 A1$="      ":A2$="      "
1370 LOCAT%=0:MOVE%=1
1380 FIN%=0
1390 CAPTURE%(1)=0:CAPTURE%(2)=0
1410 RETURN
1450 REM TITLE-SCREEN ROUTINE
1460 POKE 53280,0:POKE 53281,0
1470 PRINT"[CLEAR][DOWN][DOWN][DOWN][DOW
N]              [c 3][RVSON][c V][c I][c
 C][RVSOFF]  [RVSON][c V][c I][c C]"
1480 PRINT"                [RVSON] [RVSOFF
][c D][c I]  [RVSON] [RVSOFF] [RVSON] "
1490 PRINT"                [RVSON] [RVSOFF
] [RVSON] [RVSOFF]  [RVSON] [RVSOFF] [RV
SON] "
1500 PRINT"                [c C][RVSON][c
 I][RVSOFF][c V]  [c C][RVSON][c I][RVSOF
F][c V]"
1510 PRINT"[DOWN][DOWN]          [CYAN]BY
 MARCUS  JEFFERY"
1520 PRINT"[DOWN][PURPLE]YOU WILL PLAY T
HE [WHITE]WHITE [PURPLE]STONES,  AND"
1530 PRINT"THE  COMPUTER  (BEING WEAKER!
)  WILL"
1540 PRINT"HAVE THE [c 3]RED [PURPLE]STO
NES WITH A  HANDICAP"
1550 PRINT"              ADVANTAGE."
1560 IP%=20:IM%=7:IW%=1:GOSUB 1990
1565 HND%=VAL(A$)
1570 IF HND%<2 OR HND%>9 GOTO 1560
1590 RETURN
1730 REM PRINT-BOARD ROUTINE
1750 PRINT"[HOME][YELLOW]      STONES CA
PTURED BY:";
1760 PRINT"[WHITE]       MOVE"
1770 PRINT"[YELLOW]    WHITE =[CYAN] ";C
APTURE%(2);
1780 PRINT TAB(18);"[YELLOW]BLACK =[CYAN
] ";CAPTURE%(1);
1790 PRINT TAB(31);"[WHITE]";MOVE%
1800 PRINT TAB(12);"[DOWN]";AXIS$
1810 FOR Y=15 TO 1 STEP -1
1820 PRINT TAB(9);"[c 3]";RIGHT$(" "+STR
$(Y),2);" ";
1830 FOR X=1 TO 15
1840 PP%=PEEK(BOARD+16*Y+X)
1850 IF PP%=1 THEN PRINT"[c 3][s Q]";:GO
TO 1880
1860 IF PP%=2 THEN PRINT"[WHITE][s Q]";:
GOTO 1880
1870 PRINT"[c 6][s +]";
1880 NEXT
1890 PRINT"[c 3] ";RIGHT$(" "+STR$(Y),2)
1900 NEXT
1910 PRINT TAB(12);AXIS$
1920 PRINT"[DOWN][YELLOW]    LAST MOVE:";
1930 ITCP%=LOCAT%:GOSUB 2260:PRINT TAB(2
4);A1$
1940 PRINT TAB(18);"[DOWN][CYAN]";A2$
1950 RETURN
1990 REM INPUT ROUTINE
2000 IS%=0
2010 POKE 780,0:POKE 781,IP%+1:POKE 782,
39:SYS 65520
2015 FOR II=1 TO 79:PRINT CHR$(20);:NEXT
2020 A$=""
2030 PRINT"[WHITE]   ";MESS$(IM%);" ";
2060 GET I$:IF I$="" GOTO 2060
2065 IF ASC(I$)=13 GOTO 2120
2070 IF ASC(I$)<>20 GOTO 2100
2080 IF IS%>0 THEN IS%=IS%-1:A$=LEFT$(A$
,IS%):PRINT I$;:GOTO 2060
2085 GOTO 2060
2100 IF IS%<IW% THEN IS%=IS%+1:A$=A$+I$:
PRINT"[CYAN]";I$;
2110 GOTO 2060
2120 RETURN
2160 REM MESSAGE ROUTINE
2190 POKE 780,0:POKE 781,MP%:POKE 782,39
:SYS 65520
2195 FOR ML=1 TO 39:PRINT CHR$(20);:NEXT
2200 PRINT"    [RVSON][c 3]";MESS$(MM%);
O$;"[RVSOFF]";
2220 RETURN
2260 REM INT-TO-CHAR ROUTINE
2270 IF ITCP%=0 THEN PRINT"[CYAN]HANDICA
P";:RETURN
2275 C$=CHR$(ITCP%-16*INT(ITCP%/16)+64)+
MID$(STR$(INT(ITCP%/16)),2)
2280 PRINT"[CYAN]";C$;"      ";:RETURN
```

Schneider CPC 464/ 664

```
10 MEMORY &9FFF
20 DIM s(300):REM stack
30 GOSUB 170:REM init
40 GOSUB 1270:REM introduction
50 CLS:GOSUB 1730:REM print board
60 mve%=mve%+1:REM white moves here
70 IF over% THEN 60
80 mve%=mve%+1:REM black moves here
90 IF NOT over% THEN 60
100 ip%=23:im%=9:iw%=1:GOSUB 1990
110 IF a$="Y" THEN 40
115 IF a$<>"N" THEN 100
120 mp%=25:mm%=5:o$="":GOSUB 216
0:REM message
130 END
170 REM init routine
190 black%=1:white%=2:colour%=3
195 INK 1,26:REM white
197 INK 2,24:REM yellow
200 marker%=4:liberty%=8
220 board=&A000
240 DIM capture%(2)
250 axis$="A B C D E F G H I J K
 L M N O"
260 GOSUB 390:REM read messages
290 DIM dir%(4)
300 RESTORE 340
310 FOR i%=1 TO 4
320 READ dir%(i%)
330 NEXT i%
340 DATA 16,1,-16,-1
350 RETURN
390 REM read message routine
410 RESTORE 460
420 DIM mess$(9)
430 FOR m%=0 TO 9
440 READ mess$(m%)
450 NEXT m%
460 DATA "O.K. THINKING..."
470 DATA "Illegal entry: "
480 DATA "Stone already on point
: "
490 DATA "Illegal. Ko on point: "
500 DATA "Illegal. Suicide on
point: "
510 DATA "     O.K. GAME OVER"
520 DATA ""
530 DATA "How many handicap ston
es may I          have (2-9) ?"
540 DATA "Type your move (eg. H8
) PASS or          QUIT. :"
550 DATA "Do you want to play ag
ain (Y/N)?"
560 RETURN
1270 REM introduction routine
1280 GOSUB 1340:REM game init
1290 GOSUB 1450:REM title screen
1300 RETURN
1340 REM game init routine
1345 stack%=1
1360 atari1$="      ":atari2$="
1370 location%=0:mve%=1
1380 over%=0
1390 capture%(1)=0:capture%(2)=0
1410 RETURN
1450 REM title screen routine
1460 CLS
1462 FOR c=1 TO 20 STEP 3
1465 RESTORE 1474
1470 MOVE 260+c,320+c
1472 FOR i=1 TO 8:READ x,y:DRAWR
 x,y,3:NEXT i
1474 DATA -10,10,-50,0,-10,-10,0
,-80,10,-10,50,0,10,10,0,20
1476 MOVER -10,0:DRAWR 20,0
1478 MOVE 380+c,320+c
1480 FOR i=1 TO 8:READ x,y:DRAWR
 x,y:NEXT i
1482 DATA -10,10,-50,0,-10,-10,0
,-80,10,-10,50,0,10,10,0,80
1484 NEXT c
1500 LOCATE 12,13:PEN 3:PRINT
1510 LOCATE 1,16:PEN 2:PRINT "Yo
u will play the ";
1520 PEN 1:PRINT"white ";:PEN 2:
PRINT"stones,  and"
1530 PRINT"and the computer (bei
ng weaker) will"
1540 PRINT"have the ";:PEN 3:PRI
NT"red ";:PEN 2:PRINT"stones wit
```

```
h a  handicap"
1550 PRINT TAB(13)"advantage"
1560 ip%=22:im%=7:iw%=1:GOSUB 19
90:REM input
1565 hand%=ASC(a$)-48
1570 IF hand%<2 OR hand%>9 THEN
1560
1590 RETURN
1730 REM print board routine
1735 INK 0,4:BORDER 9:REM magent
a/green
1750 LOCATE 3,1:PEN 2:PRINT" Sto
nes captured by:";TAB(32)"Move"
1770 PRINT TAB(3)"White =";:PEN
3:PRINT capture%(2);
1780 PEN 2:PRINT"   Black =";:PE
N 3:PRINT capture%(1);
1790 PRINT TAB(33)mve%
1800 PRINT:PRINT TAB(5)axis$
1810 FOR y%=15 TO 1 STEP -1
1820 LOCATE 2,20-y%:PRINT RIGHT$
(" "+STR$(y%),2);
1830 FOR x%=1 TO 15
1835 PRINT" ";
1840 p%=PEEK(board+16*y%+x%)
1850 IF p%=1 THEN PEN 3:PRINT"O"
;:GOTO 1880
1860 IF p%=2 THEN PEN 1:PRINT"O"
;:GOTO 1880
1870 PEN 2:PRINT"+";
1880 NEXT x%
1890 PEN 3:PRINT TAB(34)y%
1900 NEXT y%
1910 PRINT TAB(5)axis$
1920 PEN 2:PRINT TAB(3)"My last
move was:";
1930 itcp%=location%:GOSUB 2260:
PEN 1:PRINT TAB(30) atari1$
1940 LOCATE 16,23:PRINT atari2$
1950 RETURN
1990 REM input routine
2000 is%=0
2010 LOCATE 3,ip%:PRINT SPACE$(62)
2020 a$=""
2030 LOCATE 3,ip%:PEN 3:PRINT me
ss$(im%);
2060 i$="":WHILE i$="":i$=INKEY$
:WEND
2065 IF i$=CHR$(13) THEN 2120
2070 IF i$<>CHR$(127) THEN 2090
2080 IF is%>0 THEN is%=is%-1:a$=
LEFT$(a$,is%):PRINT CHR$(8);" ";
CHR$(8);
2085 GOTO 2060
2090 IF i$>="a" AND i$<="z" THEN
 i$=CHR$(ASC(i$)-32)
2100 IF is%<iw% THEN is%=is%+1:a
$=a$+i$:PEN 2:PRINT i$;
2110 GOTO 2060
2120 RETURN
2160 REM message routine
2190 LOCATE 3,mp%:PRINT SPACE$(3
6);
2200 LOCATE 3,mp%:PEN 3:PRINT me
ss$(mm%);o$
2220 RETURN
2260 REM int-to-char routine
2270 IF itcp%=0 THEN PEN 1:PRINT
"Handicap";:RETURN
2275 c$=CHR$(itcp%-16*INT(itcp%/
16)+64):r$=MID$(STR$(INT(itcp%/1
6)),2)
2280 PEN 1:PRINT c$;r$;:RETURN
```

Sinclair Spectrum

```
  10 CLEAR 63999
  20 DIM s(300)
  30 PRINT AT 10,10;"PLEASE WAIT
": GO SUB 170
  40 GO SUB 1270
  50 CLS : GO SUB 1730
  60 LET move=move+1: REM white
moves here
  70 IF end THEN  GO TO 100
  80 LET move=move+1: REM black
moves here
  90 IF NOT end THEN  GO TO 60
 100 LET ip=20: LET im=10: LET i
w=1: GO SUB 1990
 110 IF a$="Y" THEN  GO TO 40
 115 IF a$<>"N" THEN  GO TO 100
 120 LET mp=21: LET mm=6: LET o$
="": GO SUB 2160
 130 STOP
 170 REM initialise routine
 190 LET black=1: LET white=2: L
ET colour=3
 200 LET marker=4: LET liberty=8
 220 LET board=64000
 240 DIM c(2)
 260 GO SUB 390
 290 DIM d(4)
 300 RESTORE 340
 310 FOR l=1 TO 4
 320 READ d(l)
 330 NEXT l
 340 DATA 16,1,-16,-1
 350 RETURN
 390 REM read-messages routine
 410 RESTORE 460
 420 DIM m$(10,43)
 430 FOR m=1 TO 10
 440 READ m$(m)
 450 NEXT m
 460 DATA "O.K. THINKING..."
 470 DATA "Illegal entry: "
 480 DATA "Stone already on poin
t: "
 490 DATA "Illegal. Ko on point:"
 500 DATA "Illegal. Suicide on p
oint: "
 510 DATA "    O.K. GAME OVER."
 520 DATA ""
 530 DATA "How many handicap sto
nes may I  have (2-9) ?"
 540 DATA "Type your move (eg. H
8), PASS   or QUIT :"
 550 DATA "Do you want to play a
gain       (Y/N) ?"
 560 RETURN
1260:
1270 REM introduction routine
1280 GO SUB 1340
1290 GO SUB 1450
1300 RETURN
1340 REM game-init routine
1345 LET stack=1
1360 LET x$="      ": LET y$="
  "
1370 LET location=0: LET move=1
1380 LET end=0
1390 LET c(1)=0: LET c(2)=0
1410 RETURN
1450 REM title-screen routine
1460 BORDER 0: PAPER 0: CLS
1470 INK 2: PRINT AT 3,12;"sh23s
h1  sh23sh1"
1480 PRINT AT 4,12;"sh84sh3  sh8
 sh8"
1490 PRINT AT 5,12;"sh8 sh8  sh8
 sh8"
1500 PRINT AT 6,12;"132  132"
1510 PRINT AT 8,7; INK 5;"by  Ma
rcus Jeffery"
1520 PRINT : PRINT  INK 3;"You w
ill play the "; INK 7;"white"; I
NK 3;" stones."
1530 PRINT  INK 3;"and the compu
ter (being weaker)"
1540 PRINT  INK 3;"will have the
 "; INK 2;" red  "; INK 3;"stone
s with"
1550 PRINT  INK 3;"      a handic
ap advantage."
1560 LET ip=15: LET im=8: LET iw
=1: GO SUB 1990
1565 LET hand=CODE (a$)-48
1570 IF hand<2 OR hand>9 THEN  G
O TO 1560
1590 RETURN
1730 REM print-board routine
1750 PRINT AT 0,0; INK 6;" Stone
s captured by:";
1760 PRINT  INK 7;"       Move "
1770 PRINT  INK 6;" White="; INK
 5;c(2);
1780 PRINT TAB 10; INK 6;"  Blac
k ="; INK 5;c(1);
1790 PRINT TAB 27; INK 7;move
1810 FOR y=15 TO 1 STEP -1
1820 PRINT AT 17-y,5; INK 2;(" "
+STR$ y)(LEN STR$ y TO );" ";
1830 FOR x=1 TO 15
1840 LET pp=PEEK (board+16*y+x)
1850 IF pp=1 THEN  PRINT  PAPER
4; INK 2;"O";: GO TO 1880
1860 IF pp=2 THEN  PRINT  PAPER
4; INK 7;"O";: GO TO 1880
1870 PRINT  PAPER 4; INK 0;"+";
1880 NEXT x
1890 PRINT  INK 2;" ";y
1900 NEXT y
1910 PRINT TAB 8; INK 2;"ABCDEFG
HIJKLMNO"
1920 PRINT  INK 6;"Last move: ";
1930 LET itcp=location: GO SUB 2
260: PRINT TAB 20; INK 5;x$
1940 PRINT AT 20,20; INK 5;y$: B
EEP 1,0
1950 RETURN
1990 REM input routine
2000 LET is=0
2010 PRINT AT ip,0;: FOR i=1 TO
62: PRINT " ";: NEXT i
2020 LET a$=""
2030 PRINT AT ip,0; INK 7;m$(im)
;
2060 LET i$=INKEY$: IF i$="" THE
N  GO TO 2060
2065 IF CODE i$=13 THEN  GO TO 2
120
2070 IF CODE i$<>12 THEN  GO TO
2090
2080 IF is>0 THEN  LET is=is-1:
LET a$=a$(1 TO LEN a$-1): PRINT
CHR$ 8;" ";CHR$ 8;: GO TO 2060
2085 GO TO 2060
2090 IF i$>="a" AND i$<="z" THEN
  LET i$=CHR$ (CODE i$-32)
2100 IF is<iw THEN  LET is=is+1:
 LET a$=a$+i$: PRINT  INK 5;i$;
2110 GO TO 2060
2120 RETURN
2160 REM message routine
2190 PRINT AT mp,0;: FOR m=1 TO
30: PRINT " ";: NEXT m
2200 PRINT AT mp,0; INK 2;m$(mm,
 TO 26);o$;
2220 RETURN
2260 REM int-to-char routine
2270 IF itcp=0 THEN  PRINT  INK
5;"Handicap";: RETURN
2275 LET c$=CHR$ (itcp-16*INT (i
tcp/16)+64): LET r$=STR$ (INT (i
tcp/16))
2280 PRINT  INK 5;c$;r$;: RETURN
```

Von Baum zu Baum

Wir erweitern unser Programm um einige Sortierroutinen, die Bäume der unterschiedlichsten Strukturen handhaben können.

Bäume mit unregelmäßigen internen Strukturen – die beispielsweise bei einem Datenpunkt drei Verzweigungen aufweisen, bei einem anderen zwei, bei einem dritten vier, etc. – sind problematisch. Die wesentliche Schwierigkeit liegt in der Prüfung, ob der Endpunkt einer Verzweigung erreicht wurde oder nicht. Bei unserem Baum zur Objektsteuerung konnten wir die Datenpunkte wegen seiner internen Struktur in verschiedene Kategorien unterteilen: zuerst die Verzweigungspunkte, dann die Schnittpunkte, die aus dem Baum oder in ihn hinein führen, danach die Endpunkte, die spezielle Abläufe oder Meldungen veranlassen, und schließlich die Endpunkte, die nur einen Rücksprung auslösten. Beim Durchgang durch den Baum (Zeilen 5030 bis 5090) konnten wir einfach auf die Routine mit der Kategorienummer des Datenpunktes verzweigen.

Leider läßt sich diese Methode bei einigen der gezeigten Bäume nicht anwenden.

Wir brauchen ein sicheres Sortiersystem, das auf jeden Baum anwendbar ist – unabhängig von seiner inneren Struktur.

Hier die Lösung: Wir müssen zunächst die Variablen des Spielablaufs initialisieren, die bisher noch nicht verwandt wurden. Tauschen Sie dazu Zeile 190 aus gegen:

```
190 DIM t(5,25,4),k(3,30),c(25),s(6),h(6
): z=0
```

Die Zeile erweitert das Array t, um die neuen Bäume bearbeiten zu können, und gibt Array c mehr Platz für neue Testbedingungen. Die Arrays s und h zeigen an, ob eine Spielfigur das Opfer gesehen und die leere Konservenbüchse gefunden hat. Die Variable z vermerkt, ob es bereits ein Opfer gibt. (Im Verlauf des Spiels wird z auf die Figurnummer des Opfers gesetzt.)

Der Baum wird programmiert: In den vier Elementen der untersten Dimension des t-Arrays werden die Informationen der Datentyptabelle über die einzelnen Datenpunkte gespeichert. Dadurch läßt sich bei der Sortierung des Baumes leicht der Datentyp jedes Elementes feststellen und zu der entsprechenden Routine verzweigen. Zuerst sollten die Daten aus Listing 1 eingegeben und mit folgenden Zeilen in das Array t übertragen werden:

```
230 REM plot tree
240 FOR n=1 TO 22: FOR s=1 TO 4: READ t(
2,n,s): NEXT s: READ a$: NEXT n
```

Beachten Sie, daß die Leerzeichen in den Zeilen 6270 und 6280 nur eingefügt wurden, um die Datengruppen besser kenntlich zu machen. Sie werden in Zeile 240 in die temporäre Variable a$ eingelesen.

Das „Sortiermodul" des Baums (Zeilen 5400 bis 5550 – Listing 2) steuert die entsprechenden Routinen an. Der Ablauf ist einfach: Zeile 5470 stellt den Typ des Datenpunktes fest und addiert Eins dazu. Über die Ergebniszahl (zwischen 1 und 7) wird dann auf eine der angegebenen Zeilen verzweigt. Ein einfacher Verzweigungspunkt mit nur zwei Alternativen (Typ 0) führt zu Zeile 5480, wo mit dem Wert der in t(2,Datenelementnummer,2) gespeicherten Bedingung entschieden wird, ob eine Verzweigung zu t(2,Datenelementnummer,3) oder t(2,Datenelementnummer,4) erfolgen soll.

Die übrigen Datenpunkttypen haben eigene Programmzeilen und holen sich ihre Parameter aus dem Array t.

Im Baumdiagramm des Spielablaufs sehen Sie, daß etliche neue Bedingungen getestet werden, die noch nicht in das Array c eingetragen wurden. Hier die Programmzeilen:

```
2450 c(13)=ABS(z=c): c(14)=ABS(g=c): c(1
5)=ABS(z=0): c(16)=ABS(s(c)=255): c(17)=
ABS(FNc(z,2)=FNc(c,2)): c(18)=ABS(h(c)=2
55): c(19)=ABS(i=2)
```

Nach Eingabe der Listings 3, 4 und 5 können Sie diesen Programmteil testen. Sie müssen nur die Steuerschleife wie in den folgenden Programmzeilen ändern:

Chancenberechnung

Punkt	Datentyp	Daten von t(Baumnummer, Elementnummer, 1–4) 1	2	3	4
0	Verzweigungspunkt	0	Bedingungsnr.	Punkt zur Verzweigung bei FALSCH	WAHR
1	GOSUB-Punkt (Baumstruktur verlassen und danach wieder zurück)	1	0	Rücksprungspunkt	Routinennummer
2	Handlungspunkt (Endpunkt, der eine Routine aktiviert)	2	0	Routinennummer	Meldungsnummer – falls vorh.
3	Meldungspunkt (Endpunkt, gibt Meldung aus)	3	0	0	Meldungsnummer
4	Endpunkt (Baum verlassen, keine Handlung)	4	0	0	0
5	Zufallspunkt (generiert Zufalls-Offset und addiert ihn zum Basispunkt)	5	0	Basispunkt	Maximaler Offset
6	Mehrfachverzweigung (Bedingungswert zum Basispunkt addieren)	6	Bedingung	Basispunkt	0

```
500 REM
510 REM test program loop
520 REM
530 GOSUB 2100: GOSUB 2150: GOSUB 2240:
PRINT: PRINT: GOSUB 1000: GOTO 530
```

Löschen Sie nun die Zeilen 540 bis 820, mit denen der Baum zur Objektbewegung getestet wurde, und geben Sie Listing 6 ein. Der Code prüft die Flags für Bearbeitung und Bewegung und initialisiert die Bedingungen.

Listing 1

Diese Programmzeilen enthalten die Daten für die Datenpunkte. Die Leerzeichen können fortfallen.

```
6240 REM
6250 REM plot tree data
6260 REM
6270 DATA 0,13,2,22," ",0,14,5,3," ",0,1
5,21,4," ",5,0,18,3," ",0,16,6,7," ",0,1
7,7,11," ",0,18,9,8," ",0,17,12,13," ",0
,19,10,17," ",5,0,14,3," ",1,0,7,1," ",4
,0,0,0," ",2,0,1,0," ",2,0,5,1," ",4,0,0
,0," ",4,0,0,0,0," ",2,0,2,4," ",2,0,3,2,"
 ",2,0,4,3
6280 DATA " ",2,0,4,0," ",4,0,0,0," ",4,
0,0,0," "
```

Listing 2

Diese Zeilen durchlaufen den Baum, testen dabei den Typ jedes Datenpunktes und verzweigen entsprechend.

```
5440 REM
5450 REM sort trees
5460 n=1
5470 ON (t(t,n,1)+1) GOTO 5480,5490,5500
,5520,5530,5540,5550
5480 k=c(t(t,n,2))+1: n=t(t,n,2+k): GOTO
 5470
5490 GOSUB 4530 : n=t(t,n,3): GOTO 5470
5500 GOSUB 4570
5510 RETURN
5520 GOSUB 4680: GOSUB 4630: GOSUB 4720
5530 RETURN
5540 a=t(t,n,4): GOSUB 4350: n=t(t,n,3)+
v: GOTO 5470
5550 k=c(t(t,n,2)): n=t(t,n,3)+k: GOTO 5
470
```

Aktionsbereit
Die Zeilen 2810 bis 2929 bewegen die Figuren von Raum zu Raum. Zeilen 3000 bis 3110 bilden den ersten Teil der Aktionstabelle, die die Routinen der Datenpunkte vom Typ 2 über die Sprungtabelle in Zeile 4510 bis 4570 aufruft. Die Routinen für die GOSUB-Punkte beginnen in Zeile 3900.

Listing 3

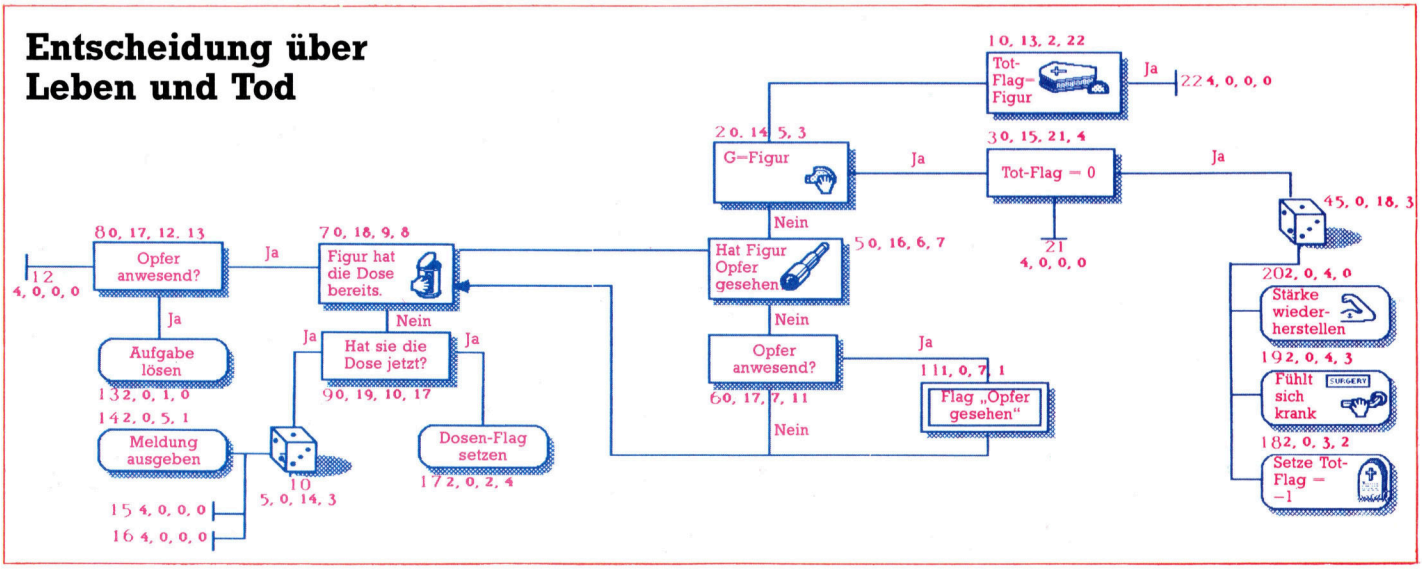
```
2810 REM
2820 REM move a character
2830 REM
2840 IF FNc(c,4)<1 THEN RETURN: REM too
weak to move
2850 y=0: f=0: FOR w=2 TO 5: IF 1$(VAL(c
$(c,2)),w)="0" THEN GOTO 2910
2860 GOSUB 4180: IF q=1 THEN f=1: GOTO 2
880
2870 GOTO 2910
2880 IF FNc(c,2)=r THEN PRINT c$(c,1);"
leaves the room...";: y=1
2890 c$(c,2)=1$(VAL(c$(c,2)),w): w=5: IF
 FNc(c,2)=r THEN PRINT c$(c,1);" enters
the room...";: y=1
2900 IF y=1 THEN y=c: GOSUB 2250: c=y: P
RINT: PRINT: REM update characters prese
nt message
2910 NEXT w: IF f=0 GOTO 2850
2920 RETURN
3000 REM
3010 REM action table
3020 REM
```

Nützliche Routinen
Dieser Programmteil baut auf unterer Ebene einige neue Routinen auf, die variable Zufallszahlen erzeugen und Meldungen ausgeben.

Listing 4

```
4270 REM
4280 REM print his/her
4290 REM
4300 IF c$(c,7)="f" THEN m$="her ": RETU
RN
4310 m$="his ": RETURN
4320 REM
4330 REM variable random number routine
4340 REM
4350 v=INT(RND(2)*a): RETURN
4360 REM
4370 REM print he/she
4380 REM
4390 IF c$(c,7)="f" THEN PRINT "she ";:
RETURN
4400 PRINT "he ";: RETURN
4500 REM
4510 REM jumpblock
4520 REM
4530 REM re-entra[illegible]
4540 ON t(t,n,4) GOSUB 39[illegible]
4550 RETURN
```

Entscheidung über Leben und Tod



```
3030 FOR n=1 TO 3: GOSUB 4090: NEXT n: m
$=c$(c,1)+" takes another look at the bo
dy, and suddenly realises the hideous tr
uth. "+CHR$(34)+"The pasty's made of cat
-food"+CHR$(34)+" ": GOSUB 4630
3040 GOSUB 4390: m$=m$+"yells, and in a
mad rush the assembled company scramble
over the bar and attack Fred the Barman,
 who pleads with them to no avail to spa
re his miserable life.": GOSUB 4630: GOS
UB 4720
3050 FOR n=1 TO 2000: NEXT n: GOSUB 4720
: m$="...and the next day Fred the Barma
n is nowhere to be seen.  However, a num
ber  of oddly shaped pasties are availab
le to feed the ever-hungry clientele of
the Dog and Bucket...": GOSUB 4630: GOSU
B 4720: END
3060 h(c)=255: GOSUB 4680: m$=c$(c,1)+m$
: GOSUB 4630: GOSUB 4720: RETURN
3070 z=c: GOSUB 4680: n$=c$(c,1)+m$: GOS
UB 4300: m$=n$+m$+"stomach, and immediat
ely expires.  The other characters are t
oo involved with their drinks to notice.
..": GOSUB 4630: GOSUB 4720: g=0: c$(c,4
)="-1": RETURN
3080 c$(c,4)="10": g=0: IF t(t,n,4)>0 TH
EN GOSUB 4680: m$=c$(c,1)+m$: GOSUB 4630
: GOSUB 4720
3090 RETURN
3100 a=20: GOSUB 4350: IF v<>5 THEN RETURN
3110 GOSUB 4680: GOSUB 4630: GOSUB 4720:
 RETURN
3900 REM
3910 REM gosub table
3920 REM
3930 s(c)=255: m$=c$(c,1)+" kneels down
beside the prostrate body of "+c$(z,1)+"
.  The horrible truth slowly dawns, but
the others seem too drunk to pay any imm
ediate attention...": GOSUB 4630: GOSUB
4720: RETURN
```

```
4560 REM action nodes
4570 ON t(t,n,3) GOSUB 3030,3060,3070,30
80,3100: RETURN
4600 REM
4610 REM print messages if player presen
t
4620 REM
4630 IF FNc(c,2)=r THEN PRINT m$;
4640 m$="": RETURN
4650 REM
4660 REM select a message from data stat
ement
4670 REM
4680 RESTORE 7030: FOR m=1 TO t(t,n,4):
READ m$: NEXT m: RETURN
4690 REM
4700 REM print a blank line
4710 REM
4720 IF FNc(c,2)=r THEN PRINT
4730 RETURN
```

Listing 5

Meldungen

Zeile 4680 gibt mit diesen Datenzeilen Meldungen aus. Da es auf Commodore-Computern kein RESTORE für bestimmte Zeilennummern gibt, müssen wir andere Techniken einsetzen. Mehr darüber in der nächsten Folge.

```
7000 REM
7010 REM message data
7020 REM
7030 DATA "A strange smell fills the air
...could it be the odour of Catty-Kit A
La Carte?"," suddenly collapses on the f
loor, clutching "," looks rather ill, an
d warns the others not to touch the past
y."
7040 DATA " examines the tin carefully,
and assumes a thoughtfull expression."
```

BASIC-Dialekte

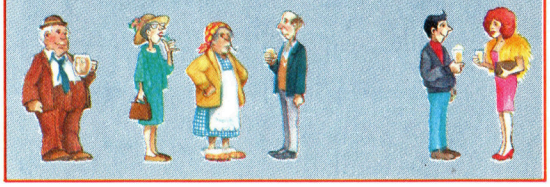
Die Programme laufen auf Schneider-Computern ohne jede Änderung. Varianten für andere Geräte erscheinen in der nächsten Folge des Computer Kurs.

Listing 6

Figursteuerung

Jede Figur wird nacheinander von der Steuerroutine angesprochen. Die Zeilen prüfen die Flags für Bewegung und Bearbeitungsart und rufen Routinen auf, die die Figur bewegen oder die Baumstruktur abfragen. Die REM-Zeilen wurden in disem Fall aus Gründen der Lesbarkeit eingefügt.

```
1000 REM
1010 REM character handler
1020 REM
1030 REM check to see if key pressed
1040 GOSUB 4260: IF i$()"" THEN GOSUB 20
40: RETURN
1050     REM process each character in t
urn
1060 FOR c=1 TO 6
1070     REM check 'handle flag'
1080 IF FNc(c,10)>0 THEN c$(c,10)=FNm$(c
$(c,10),1): GOTO 1500
1090     REM flag=0 so reset flag and pr
ocess character
1100 RESTORE: FOR n=1 TO c*10+c-1: READ
c$(c,10): NEXT n
1110 IF FNc(c,10)=0 THEN GOTO 1500: REM
default value=0 so don't process
1120     REM check move flag
1130 IF FNc(c,11)>0 THEN c$(c,11)=FNm$(c
$(c,11),1): GOTO 1190
1140     REM move flag=0 so reset flag a
nd move character
1150 RESTORE: FOR n=1 TO c*11: READ c$(c
,11): NEXT n
1160 IF c$(c,11)="0" THEN GOTO 1180
1170 GOSUB 2840: GOTO 1500
1180     REM sort through trees
1190 GOSUB 2400: REM initialise conditio
ns
1200 t=2: GOSUB 5460: REM plot tree
1210 IF FNc(c,4)>0 THEN GOSUB 5000: REM
object manipulation tree
1500 NEXT c: GOTO 1030: REM do next char
 acter - when all finished check for key
 press/do again
```



Gummiband-Betrieb

Wir schließen das Digitalisierarm-Projekt mit der zweiten Hälfte des Tracer-Programms für den Acorn B ab. Es bietet einen bequemeren Weg zur Erzeugung von Punkten, Linien, Kreisen und ausgefüllten Dreiecken. Viel Spaß beim ersten Einsatz!

Der zweite Teil des Tracer-Programms erlaubt eine Art „Gummiband"-Betrieb. Im Gegensatz zum direkten Zeichnen (Freihand-Modus) gibt man beim Gummiband-Betrieb beispielsweise nur den Anfangs- und Endpunkt einer Linie oder den Kreismittelpunkt und den Radius ein. Die fehlenden Linienzüge erzeugt das Programm dann automatisch. Im Gummiband-Betrieb erscheint oben auf dem Bildschirm ein Menü, aus dem über einfachen Tastendruck die gewünschte Funktion gewählt werden kann. Das Menü bietet auch die aus dem letzten Abschnitt bereits bekannten Möglichkeiten zum Speichern oder Laden des Bildschirminhalts. Mit Hilfe dieser Programmerweiterung können auch relativ komplexe Strukturen auf dem Bildschirm erzeugt und für die spätere Bearbeitung auf Cassette oder Diskette gespeichert werden.

Die Prozeduren

● PROCpoint

Wenn beim Aufruf von PROCdraw die Variable „togflag" auf Eins liegt, können wir den Cursor frei über den Bildschirm bewegen, ohne daß gezeichnet wird. Vor der Wahl von PROCpoint wird daher erst einmal togflag auf Eins gesetzt – also der „Stift" vom Zeichenbrett gehoben. Der Ablauf ist in eine Schleife eingebunden, die auf Tastendrücke zum Abbrechen der Schleife oder einen Wechsel der Malfarbe reagiert. Beim Druck auf den Taster am Digitalisierarm wird ein Punkt auf die aktuelle Cursor-Position gesetzt.

● PROCline

Diese Prozedur speichert zuerst die aktuellen Koordinaten des Cursors. Damit ist ein Fixpunkt des „Gummibandes" festgelegt. Von diesem Punkt aus wird zur nächsten Cursorposition eine Linie erzeugt. Das Zeichnen geschieht im Exclusive-OR-Modus mit GCOL3. Eine Linie kann wieder gelöscht werden, indem man sie mit einer neuen Linie überschreibt. Alle Funktionen sind in einer gemeinsamen Programmschleife untergebracht. Das Positionieren einer Linie wurde so realisiert,

Befehle im Tracer-Programm

Hauptmenü

Taste	Funktion
1	Freihand-Modus
2	Gummiband-Modus

Freihand-Menü

S	Bildschirminhalt auf Diskette oder Cassette speichern
L	Bildschirminhalt von Diskette oder Cassette laden
M	Rückkehr zum Hauptmenü
C	Bildschirm löschen
Taster	Stift hoch/Stift runter

Gummiband-Modus

S,M,L,C	Wie im Freihand-Modus
P	Punkt in der vorgewählten Farbe zeichnen. M führt zurück zum „Gummiband"-Menü
D	Linie von der ursprünglichen Cursorposition zur aktuellen Cursorposition ziehen. Druck auf M oder den Taster legt die Linie fest und führt ins Gummiband-Menü zurück.
F	Das durch drei Tasterbetätigungen definierte Dreieck mit Farbe ausmalen.
R	Kreis zeichnen. Der Mittelpunkt ist die Cursorposition beim Aufruf, der Radius hängt von der aktuellen Cursorposition ab. Durch Taste M oder Taster am Digitalisierarm zurück zum Gummiband-Menü.

Zeichenfarben

1	Wählt logische Farbe 1 für die Zeichnung (Vordergrund). Standardeinstellung: rot.
2	Wählt logische Farbe 2 für die Zeichnung (Vordergrund). Standardeinstellung: gelb
3	Wählt logische Farbe 3 für die Zeichnung (Vordergrund). Standardeinstellung: weiß.

Mit dem Befehl VDU 19 können die logischen Farben aus den 16 verfügbaren Farben gewählt werden. VDU 19,1,6,0,0,0 macht Cyan zur logischen Farbe 1. Die Zeichenfarben können jederzeit verändert werden.

Dieses Listing ist die Ausbaustufe des Digitalisierarm-Programms für den Acorn B. Es wird an den ersten Programmabschnitt angehängt.

als ob sie ein auf der einen Seite befestigtes Gummiband wäre, das auf der anderen aber mit dem Cursor verbunden ist. Mit ihm kann man nun über den Bildschirm wandern, bis die Linie in der gewünschten Position liegt. Mit der M-Taste oder dem Taster am Digitalisierarm wird die Linie fixiert.

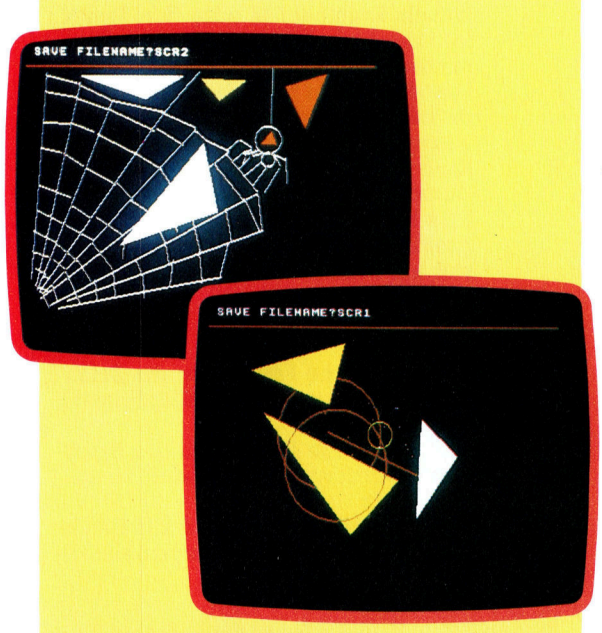
● **PROCfill**

Die Füllprozedur nutzt den Acorn B-Befehl PLOT 85,x,y, der ein Dreieck mit einer Farbe ausfüllt. Der Befehl konstruiert aus den Koordinaten x,y und den beiden zuletzt aufgesuchten Punkten ein Dreieck und färbt es vollständig ein. Auch in diesem Programmteil können die Punkte durch Druck auf die Taste am Digitalisierarm festgelegt werden.

● **PROCcircle**

Mit der letzten Programmoption können Sie den Radius eines Kreises festlegen. Für genau diesen Zweck wird PROCline in dieser Prozedur aufgerufen. Nach Festlegung des Radius springt die Programmsteuerung zu PROCcircle zurück, das zuerst die gezogene Linie und dann den Cursor selbst löscht, bevor der Kreisbogen geschlagen wird. Der Mittelpunkt wird durch die Variablen x1 und x2 definiert – den Cursorkoordinaten bei Aufruf der Routine. PROCline stellt die Koordinaten eines Punktes auf dem Kreisumfang bereit (x2,y2). Aus diesen Punkten wird der Radius berechnet. Den eigentlichen Kreisbogen erzeugt ein Standardalgorithmus.

Mit diesen Routinen wird der Digitalisierarm zu einem nützlichen Helfer.



Komfortable Bildschirmgrafik
Mit der zweiten Ausbaustufe unseres Programms für den Digitalisierarm wird das Zeichnen von geraden Linien, Kreisen und farbigen Dreiecken zu einem Kinderspiel.

Tracer-Programm Teil 2

```
 2350 DEF PROCelastic
 2360 PROCelastic_inform
 2370 REPEAT
 2380 ans$=INKEY$(1):IF ans$<>"" THEN PROCelastic_
press
 2390 togflag=1:PROCdraw
 2400 UNTIL exitflag=1
 2410 ENDPROC
 2430 DEF PROCelastic_inform
 2440 PROCcalc_xy:oldx=x:oldy=y:PROCcursor(oldx,oldy)
 2450 PROCreinform
 2460 GCOL 0,1:MOVE 0,920:DRAW 1280,920
 2470 ENDPROC
 2480 DEF PROCreinform
 2490 PRINT TAB(1,1);SPC(79)
 2500 PRINT TAB(1,1);"S=Save L=Load M=Menu C=Clear
R=Circle"
 2510 PRINT" D=Draw Line    P=Point    F=Fill"
 2520 ENDPROC
 2540 DEF PROCelastic_press
 2550 IF ans$="C" THEN CLS:PROCelastic_inform:ENDP
ROC
 2560 IF ans$="M" THEN exitflag=1
 2570 PROCcolour_change
 2580 IF ans$="R" THEN PROCcircle_title:PROCcircle
:PROCelastic_inform:ENDPROC
 L.2590,2600
 2590 IF ans$="D" THEN PROCline_title:PROCline
 2600 IF ans$="P" THEN PROCpoint_title:PROCpoint
Celastic_inform:ENDPROC
 2620 IF ans$="S" THEN PROCsave_screen:PROCelastic
_inform:ENDPROC
 2630 IF ans$="L" THEN PROCload_screen:PROCelastic
_inform:ENDPROC
 2640 ENDPROC
 2660 DEF PROCpoint_title
 2670 PRINT TAB(1,1) SPC(79)
 2680 PRINT TAB(15,1);"Point Mode"
 2690 ENDPROC
 2710 DEF PROCpoint
 2720 REPEAT
 2730 togflag=1:PROCdraw
 2740 ans$=INKEY$(1)
 2750 PROCcolour_change
 2760 IF (ADVAL(0)AND 3)<>0 THEN PLOT 69,x,y:REPEA
T UNTIL (ADVAL(0)AND 3)=0
 2770 UNTIL ans$="M"
 2780 PROCreinform
 2790 ENDPROC
 2800 DEF PROCline_title
 2810 PRINT TAB(1,1) SPC(79)
 2820 PRINT TAB(15,1);"Line Mode"
 2830 ENDPROC
 2850 DEF PROCline
 2860 x1=x:y1=y:REPEAT
 2880 PROCcalc_xy:PROCcursor(oldx,oldy)
 2890 x2=x:y2=y:GCOL 3,colour:MOVE x1,y1:DRAW x2,y2
 2900 ans$=INKEY$(1)
 2910 IF (ADVAL(0)AND 3) THEN ans$="M"
 2920 PROCcursor(x,y):oldx=x:oldy=y
 2930 IF ans$<>"M"THEN GCOL 3,colour:MOVEx1,y1:DRA
W x2,y2
 2940 PROCcolour_change
 2950 UNTIL ans$="M"
 2960 PROCreinform
 2970 ENDPROC
 2990 DEF PROCfill_title
 3000 PRINT TAB(1,1) SPC(79)
 3010 PRINT TAB(15,1);"Fill Mode"
 3020 ENDPROC
 3040 DEF PROCfill
 3050 FOR I=1 TO 3
 3060 REPEAT
 3070 togflag=1:PROCdraw:ans$=INKEY$(1)
 3075 IF ans$<>""THEN PROCcolour_change
 3080 UNTIL (ADVAL(0)AND 3)<>0
 3090 PLOT 69,x,y:x(I)=x:y(I)=y
 3110 REPEAT UNTIL (ADVAL(0)AND3)=0:REM AWAIT BUTT
ON OFF
 3120 NEXT I
 3135 PROCcursor(x,y):REM cursor off
 3140 FOR I=1 TO 2:PLOT 69,x(I),y(I):NEXT I
 3150 PLOT 85,x,y
 3160 ENDPROC
 3180 DEF PROCcircle_title
 3190 PRINT TAB(1,1) SPC(79)
 3200 PRINT TAB(15,1);"CIRCLE MODE"
 3210 ENDPROC
 3230 DEF PROCcircle
 3240 PROCline
 3250 GCOL 3,colour:MOVE x1,y1:DRAW x2,y2:REM RUBO
UT  LINE
 3260 GCOL 0,colour:REM BACK TO NORMAL PLOT MODE
 3270 PROCcursor(x2,y2)
 3370 radius=SQR((x2-x1)^2+(y2-y1)^2)
 3380 MOVE x1+radius,y1
 3390 FOR angle=0 TO 2*PI STEP 0.1
 3400 rx=x1+radius*COS(angle)
 3410 ry=y1+radius*SIN(angle)
 3420 DRAW rx,ry
 3430 NEXT angle
 3435 DRAW x1+radius,y1
 3440 ENDPROC
```

Fachwörter von A bis Z

Minicomputer = Minicomputer
Ursprünglich verstand man unter Minicomputer Rechner von der Größe eines Aktenschranks, im Unterschied zu den Mainframes, die weit mehr Platz beanspruchten. Minicomputer sind langsamer und weniger leistungsfähig als ihre großen Brüder, aber sehr viel preiswerter und daher auch schon in kleineren Betrieben anzutreffen, für deren Erfordernisse Mainframes weit überdimensioniert und zu kostspielig wären.

Einige Fachleute haben angesichts immer leistungsfähigerer Micros das Aussterben der Minicomputer prophezeit. Es scheint aber so, als ob sie sich trotz des drastischen Leistungszuwachses bei den Microprozessoren noch etliche Jahre halten würden.

Modem = Modem
Ein Modem (als Abkürzung für Modulator/Demodulator) ist ein Peripheriegerät für die Rechnerkommunikation über Fernsprechleitungen; es besorgt die Umwandlung der Digitalsignale eines Rechners in übertragbare Analogsignale und umgekehrt. Diese Umsetzung wird in der Fachsprache als Modulation bzw. Demodulation bezeichnet.

Die heute angebotenen Modems lassen sich im wesentlichen zwei Gruppen zuordnen: Bei den Akustikkopplern besteht nur eine akustische Verbindung mit dem Fernsprechnetz durch den auf das Modem gesetzten Hörer, während bei den direkt gekoppelten Modems über einen Zwischenstecker an der Telefonsteckdose ein unmittelbarer elektrischer Kontakt mit dem Postnetz hergestellt wird.

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.

Bei der Demodulation wird aus dem analogen Telefonsignal wieder ein für den Computer verständliches Bitmuster generiert. Außer für die Modulation und Demodulation sind Modems auch dafür eingerichtet, die Baudrate bei der Übertragung anzupassen. Da es bisher keinen verbindlichen Baudratenstandard gibt, sind die Übermittlungsgeschwindigkeiten oft unterschiedlich, und ein Datenaustausch zwischen Rechnern ist ohne Anpassungsvorkehrungen unmöglich.

Mit zunehmender Verbreitung von Microcomputern und Datennetzen werden immer mehr Informationssysteme für Micros mit Modem zugänglich.

Modular Programming = Modulare Programmierung
Modulare und strukturierte Programmierung sind praktisch gleichbedeutend. Gemeint ist damit die Unterteilung eines Programms in Abschnitte, von denen jeder nur für eine Aufgabe zuständig ist, im Gegensatz zum „monolithischen" Programmierstil. Die Modularität erlaubt es, die einzelnen Bausteine unabhängig voneinander zu testen, bis sie einwandfrei laufen. Dann werden sie durch Parameteraustausch untereinander verknüpft, und das Programm arbeitet als Ganzes.

Modulation = Modulation
Modulation wird in der Nachrichtentechnik das Aufprägen von Information auf ein Trägersignal genannt; beim Datenverkehr erfolgt das im allgemeinen durch Umsetzen eines Digitalsignals in ein Analogsignal. Der umgekehrte Vorgang heißt Demodulation. Es gibt verschiedene Modulationsverfahren: Bei der Amplitudenmodulation (AM) wird die Amplitude, das heißt, die Höhe einer als Träger dienenden Wechselspannung verändert, während bei der Frequenzmodulation (FM) die Frequenz des Wechselspannungsträgers variiert wird. Damit verwandt ist die Phasenmodulation (PM), bei der die Information in einer zeitlichen Verschiebung des Trägers enthalten ist. Bei Hochgeschwindigkeits-Modems wird zuweilen ein kombiniertes AM/PM-Verfahren benutzt. Die Pulscodemodulation (PCM) dient dagegen zur digitalen Darstellung eines Analogsignals, dessen jeweilige Höhe in kurzen Zeitabständen abgetastet und durch ein Impulsmuster codiert wird.

Modulus = Modul
Das Wort ‚Modul' hat mehrere Bedeutungen. Es wird unter anderem in der Mathematik zur Bezeichnung der arithmetischen Operation für das Bestimmen des Teilungsrests – etwa bei der Division A:B – verwendet. Dafür schreibt man „A mod B" (gesprochen „A modulo B"); in PASCAL findet es sich als MOD-Anweisung wieder und wird meist in Verbindung mit dem DIV-Befehl für die ganzzahlige Division (ohne Rest) benutzt. Zur Illustration: 17 DIV 5 ergibt 3, und 17 MOD 5 liefert 2, da bei der Rechnung 17:5=3 der Rest 2 herauskommt.

Bildnachweise
1849: Chris Corr
1850, 1851: Macintosh/Maccharlie
1852, 1853, 1861–1863, U3: Ian McKinnell
1858: Liz Heaney
1864, 1865: Kevin Jones
1868: Liz Dixon
1869, 1873: Caroline Clayton
U4: Rolf Seiffe

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

computer kurs Heft 68

In unserem neuen Programmier-Service bringen wir Programme, mit denen das tägliche Leben erleichtert wird. Wer fragt sich nicht immer wieder, wo nur das ganze Haushaltsgeld bleibt. Unser „Kostenkalkulator" wurde für Atari-, Commodore-, Sinclair- und Schneider-Rechner geschrieben. Er erleichtert die Übersicht über Einnahmen und Ausgaben, und man kann Kosten gering halten.

MIDI-Rock

Entwurf und Konstruktion eines MIDI-Interface für den Acorn B und C64.

Auf die Plätze

Nach der Ausarbeitung der Regeln und einiger E/A-Routinen werden nun noch kleinere Routinen integriert.

Chip-Tourismus

Eine Platinenrundreise erklärt die Funktionen einzelner Bauteile.